

AD-A124 662

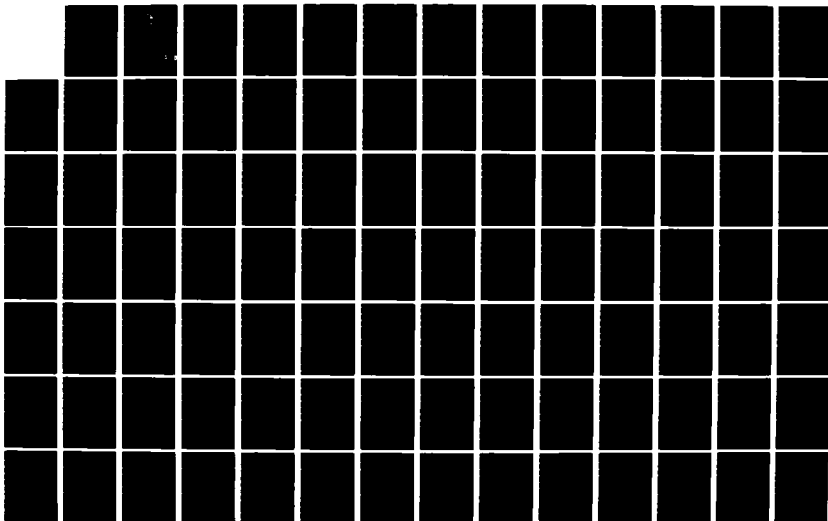
STATIC AEROELASTIC ANALYSIS OF FLEXIBLE WINGS VIA
NASTRAN PART I (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING K JONES
DEC 82 AFIT/GAE/RA/82D-16-PT-1

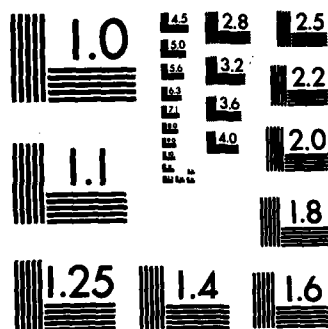
1/2

UNCLASSIFIED

F/G 1/3

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 124662



STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN

THESIS

AFIT/GAE/AA/82D-16

Kim Jones
1Lt USAF

DTIC
ELECTE
FEB 22 1983
S D E

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

DTIC FILE COPY

Wright-Patterson Air Force Base, Ohio

This document has been approved
for public release and sale; its
distribution is unlimited.

93 02 022128

①

STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN

THESIS

AFIT/GAE/AA/82D-16

Kim Jones
1Lt USAF

Approved for public release; distribution unlimited

DTIC
ELECTE
1102 2 1983
E

STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART I

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

By
Kim Jones, B.S.A.E.
1Lt USAF
Graduate Aerospace Engineering
December 1982

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	



Approved for Public Release; Distribution Unlimited

Preface

This thesis is a continuation of the research done by Captain Lance P. Chrisinger in his M.S. Thesis at AFIT in 1980. The first part contains the theoretical development of the sequence along with the results of a survey of wing models. The survey determined the characteristics of average wing models that the procedure must be able to tolerate. The second part develops a procedure for designing and testing a NASTRAN module. Then the module used for the aeroelastic sequence, GIAS, is used as an example. The logical procedures in GIAS are then explained. Finally, a detailed description of the aeroelastic sequence is given along with a description of how to implement it.

I would like to extend my gratitude to Captain Hugh C. Briggs, who has understood and put up with me while guiding me through this thesis.

I would also like to thank Michael Bernier and Jim Trace for their help with the computer.

A very special thanks goes to my wife, Valerie, whose patience and understanding has helped greatly with my stay at AFIT.

Contents

<u>Part I.</u>	<u>Page</u>
Preface	ii
List of Figures	iv
Abstract.	v
I. Introduction.	1
II. Background.	2
Dual-Program Method.	2
FLEXLOADS.	3
FLEXSTAB	3
Via NASTRAN.	4
III. Theory.	5
Iterative.	5
Noniterative	7
Comparison	9
IV. Capabilities.	11
V. Procedure	13
VI. Survey.	14
Characteristics to be Determined	14
Characteristics of Wings in Survey	16
Effects of Characteristics	25
Bibliography.	29

Part II.

I. Introduction.	1
II. Designing and Testing a NASTRAN Module.	2
III. Example of a New NASTRAN Module, GIAS	9
IV. Logical Procedures Within GIAS.	18
V. Running the Flexible Wing Solution Sequence	22

List of Figures

<u>Figure</u>		<u>Page</u>
<u>Part I.</u>		
1	Graphical Description of T-37 Wing Model	17
2	Graphical Description of T-38 Wing Model	19
3a	Graphical Description of "Eglin Wing" Model.	21
3b	Graphical Description of "Eglin Wing" Model (Wing Tip)	22
4a	Graphical Description of B-1 Wing Model (Inboard Substructure) . . .	23
4b	Graphical Description of B-1 Wing Model (Outboard Substructure) . .	24
<u>Part II.</u>		
1	Link 11	16
2	Determining AOAP.	24

Abstract

↓
The purpose of this study was to expand the capabilities of the Static Flexible Wing Aeroelastic Sequence that Captain Lance P. Chrisinger developed for NASTRAN as his Master's Thesis at AFIT. Captain Chrisinger developed a basic procedure to enable NASTRAN to analyze flexible wing airloads and stresses. That capability is expanded to enable analysis of standard wing models.

A subroutine was incorporated into NASTRAN to eliminate extensive hand-calculation of transformation matrices.

The capability to tolerate control surfaces was discovered. Also, a survey of wing models in the Air Force inventory was taken to determine the characteristics of the average wing model. This was used to determine where the aeroelastic procedure was lacking in its ability to analyze wing models.

↑

STATIC
AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART I

I. Introduction

An improved aeroelastic package has been developed to calculate stresses, displacements, and aerodynamic coefficients for flexible wings undergoing computer-generated airloads. The package is based on the readily available NASTRAN computer program and can tolerate moderately complex wing models with secondary structure. The need for such a package became evident as aircraft wings were designed with more structural flexibility. The changes in displacements caused by the airloads, and the changes in airloads caused by the displacements became significant enough to warrant attention. As a result, several aeroelastic analysis packages have been designed, such as FLEXLOADS and FLEXSTAB. However, all of the packages designed so far contain deficiencies that prevent them from efficiently solving complex wing structures. This package is an attempt to design an aeroelastic analysis program that does not have any serious drawbacks.

II. Background

The aeroelastic flexible wing problem is a study in fluid-structure interaction because the airloads on the structure are dependent on the displacements of the structure. Several packages have been designed to solve this interaction problem for aircraft wings.

Dual-Program Method

One type of solution technique uses an airloads generator computer program and a structural analysis package. The advantage of this method is that its parts are already in existence. However, there are some serious drawbacks to this solution technique. Much time is consumed in transforming the generated airloads into forces for the structural analysis package. Also, the displacements from the structural analysis package must be transformed into displacements for the airload generator. An additional problem with this method is the duplicate computer work required. As iterative passes are made through the airloads generator and structural analysis package the same equations are assembled and decomposed. This results in the same work being done with each pass through the system.

FLEXLOADS

Another solution technique aimed at solving the flexible wing problem involves a single program using an iterative solution. The program, FLEXLOADS, was developed by General Dynamics where it is currently being used to analyze several bombers and transports. They are also modifying FLEXLOADS to use with nonlinear aero theory in order to investigate the separation and stall flight regimes. Northrup is using FLEXLOADS in their analysis of several fighters. The advantages of FLEXLOADS are that it is a relatively small, easily manageable program and that it contains several different aero theories to choose from. A disadvantage is its size restriction of 165 degrees-of-freedom (DOF), which will limit the complexity of the models that can be analyzed. An attempt is being made at San Antonio Air Logistics Center (SAALC) to use Guyan Reduction to reduce the number of DOF of a T-37 wing model from 850 to 165 DOF and still keep its accuracy. They are currently having difficulty finding a representative set of nodes that leaves the natural frequencies and static displacements unaltered.

FLEXSTAB

Still another way to solve the flexible wing problem is to use FLEXSTAB, a computer program developed by Boeing. FLEXSTAB was originally designed as a stability analysis package but can also analyze structures. It is currently being used at Nasa-Dryden and Nasa-Wallops. The primary

disadvantage of FLEXSTAB is that it gives only approximations of the stresses and displacements because it uses equivalent beam elements rather than the detailed model.

Via NASTRAN

Captain Lance Chrisinger, in his Master's Thesis at AFIT, had previously done all the preliminary work of comparing these various solution techniques (Ref 3). The problems a solution technique must be able to tolerate in order to effectively solve a flexible wing are: a wide variety of element types, any odd characteristics of typical wing models such as node number resequencing (SEQGP) cards, and large numbers of elements. After looking at these requirements and the characteristics of all the available solution techniques it became apparent that there is a need for yet another solution technique. The solution technique must involve a complete, self-contained computer program. All of these qualities were found in the NASTRAN computer program. The one disadvantage that this program does have is that it is rather large and unwieldy. However, with the recent inclusion of an airload generator, NASTRAN has all the requirements to effectively solve a flexible wing. In addition, Captain Chrisinger built an instruction sequence for solving the flexible wing problem for a simple generic wing box (Ref 3).

III. Theory

There are two approaches to solving flexible wings. Both are based on the same assumptions but differ in methodology. In one scheme iterative passes are made through a sequence of steps while in the other the process is assembled into a single equation to be solved (Ref 2).

Iterative

The iterative solution consists of three parts: a pre-processor to set up the operations, an iterative loop that terminates when a specified tolerance is reached, and a post-processor to recover the output.

Definitions

- * - general matrix multiplier
- U_s - the change in structural deformation
(the first value is the initial structural displacements representing the initial angle-of-attack)
- U_{ST} - The cumulative structural deformation
- Q_s - The change in forces on the structural grid
- Q_{ST} - The cumulative forces on the structural grid
- Q_A - The changes in airloads on the aero grid
- Q_{AT} - The cumulative airloads on the aero grid

Procedure

```

set Us = Angle of Attack      initial angle of
set Usr = - Angle of Attack   attack so the
                                initial angle of
                                attack will not be
                                included in summing
                                the deformations

```

set $Q_{ST} = 0$

set $Q_{AT} = 0$

top of loop

$$U_A = G * U_S$$

G - transformation
matrix from struc-
tural displacements
to aero displace-
ments

$$Q_A = A_{JJ}^{-1} * U_A$$

A_{α} - matrix of aero coefficients

$$Q_S = G' * Q_A$$

G' - transformation
matrix from airloads
on aero grid to
forces on structural
grid

$$U_s = K^{-1} * Q_s$$

K - stiffness matrix
cumulating structural
deformation
changes

$$U_{ST} = U_{ST} + U_S$$

$Q_{ST} = Q_{ST} + Q_s$ cumulating changes
in forces on struc-
tural grid

$Q_{AT} = Q_{AT} + Q_A$ cumulating changes
in airloads on aero
grid

go to top of loop unless the norm of the change
in structural displacements is less than
the tolerance

output

U_{ST}

Q_{ST}

Q_{AT}

aerodynamic coefficients

Noniterative

The noniterative scheme uses the same assumptions
but it is ordered differently (Ref 2):

Definitions

- * - general matrix multiplier
- K - the stiffness matrix
- U_s - structural deformations
- Q_s - structural forces
- G - the transformation from aero forces to
structural forces and also from structural
deformations to aero deformations
- AOA - initial angle of attack
- U_A - aero deformations

Q_A - aero forces
 CQ_A - coefficients of aero forces
 CP_A - coefficients of aero pressures
 $D1JK$ - matrix of aero coefficients, real
 part of complex aero wash
 $D2JK$ - matrix of aero coefficients, imaginary
 part of complex aero wash
 A_{JJ}^{-1} - matrix of aero influence coefficients
 SKJ - matrix of areas of the wing
 Q - dynamic pressure

Procedure

$$\begin{aligned}
 U_A &= G^T * U_s \\
 CP_A &= A_{JJ}^{-1} * [D1JK + i(D2JK)] * U_A \\
 CQ_A &= SKJ * CP_A \\
 Q_s &= Q * G * CQ_A
 \end{aligned}$$

therefore

$$Q_s = Q * G * SKJ * A_{JJ}^{-1} * [D1JK + i(D2JK)] * G^T * U_s$$

but

$$D2JK = 0 \quad \text{only concerned with static, zero frequency}$$

and

$$F = G * SKJ * A_{JJ}^{-1} * AOA \quad \text{rigid body } Q_s \text{ due to AOA}$$

and

$$FDEF = G * SKJ * A_{JJ}^{-1} * xD1JK * G^T \quad Q_s \text{ due to deformations}$$

therefore

$$Q_s = Q * [(F * AOA) + (FDEF * U_s)]$$

if

$$U_s = K^{-1} * Q_s$$

then

$$U_s = Q * K^{-1} * (AOA * F) + (FDEF * U_s)$$

$$[1 - (Q * K^{-1} * FDEF) * U_s = Q * K^{-1} * F * AOA$$

finally

$$U_s = [1 - (Q * K^{-1} * FDEF)]^{-1} * Q * K * F * AOA$$

The last equation is the one that is used for the noniterative scheme.

Comparison

There are two points to be considered when deciding which solution scheme to use for a particular problem. Both the accuracy and the expense (in terms of time to compute) is discussed. The accuracy of the iterative solution scheme can, at best, equal the accuracy of the noniterative solution. This is because they both make the same approximations but the iterative solution has an additional inherent error. The iterative solution takes a shorter time to assemble and decompose than the noniterative scheme but then also involves the time in going through the iteration loop. The noniterative scheme takes longer to assemble and decompose but once this is done, the equations are solved only once. Therefore, if the time to assemble and decompose the equations for a particular model in the iterative solution is significantly

shorter than that for the noniterative solution, then the iterative solution will take less time. In order for this to work, the assembly and decomposition time for the iterative solution has to be shorter than the noniterative solution by the amount of time it takes for the iterative solution to complete all the iterative passes. Recent experience indicates typically 3-6 passes are required for simple models. For the present work, the iterative scheme has been employed.

IV. Capabilities

The discussion of the capabilities of the current solution technique is broken into two parts. The first gives an outline of the applicable characteristics of NASTRAN. The second describes the capabilities of the instruction sequence.

As outlined in the background section, the primary disadvantage in using a program such as NASTRAN is its large size. Because of this the instruction sequence is more difficult to manipulate. Another disadvantage is that NASTRAN uses only one aero theory, a Doublet-Lattice Method. This theory does not generate any rotational forces; therefore, any bending elements in the model will not have forces along their rotational DOF due to the airloads. Despite these disadvantages, NASTRAN was chosen as a host program because of its several advantages. The most important of these is NASTRAN's ability to tolerate very large problems efficiently. Another major advantage is the wide variety of element types already available in NASTRAN. In addition, many options to the instruction sequence are easily implemented. These three advantages lend a great deal of flexibility to any instruction

sequence implemented with NASTRAN. This is essential when dealing with the wide range and variety of wing characteristics encountered.

This second section outlines the capabilities of the instruction sequence. The ultimate goal for the sequence is to be able to solve for the stresses, displacements, and aerodynamic coefficients of an entire flexible aircraft. Currently, a restricted type of flexible wing can be solved. Provisions have been made to calculate internally all of the transformation matrices and other derived input data. This enables the user to analyze a wing using an existing model with minimum additional input data. The instruction sequence is capable of solving deflected control surfaces as long as the deflections are small enough to stay within linear aero theory. Substructuring has not been attempted while using the instruction sequence. The only wing surface element type that can be used by the instruction sequence is the CQDMEM2 element. Provisions have yet to be made to tolerate grid point number resequencing (SEQGP) cards. Wing-fuselage aerodynamic interaction can be modeled by constructing an aero model of the fuselage near the wing root.

V. Procedure

This section is included in the event that someone with an understanding of the instruction sequence has a desire to use it. Within this section is an overview of additional input data that must be included with a standard model in NASTRAN format. The details of each additional piece of data are covered in Part II.

The instruction sequence requires that two subcases be run. The first subcase generates the initial angle of attack and the second subcase actually goes through the iterations. There are several PARAM cards that must be added. These are concerned with various dimensions of the wing model, the angle of attack, and the error deemed adequate for the iteration convergence. SPC cards are required for both subcases to help determine the AOA and to partition out the rotational DOF of the model. ASET cards are needed to narrow down the analysis set to just the Z DOF of every free node. A CAERO card and its support cards are needed to determine the flight regime. An EIGR card is needed to help set the initial angle of attack. SPLINE and SET cards are needed to build the transformation matrix from the displacements on the structural grid to the displacements and slopes on the aero grid. Finally, a table (TOSEL - Top Surface Element List) must be input in the Bulk Data Deck.

VI. Survey

In order for the instruction sequence to run efficiently with standard wing models their characteristics must be known. To this effect a survey of several wing models in the Air Force inventory was taken. The answers to a list of questions were compiled to determine the characteristics of the average wing model. Though several wings were surveyed, only four will be presented here. These depict typical wing models. The rest are not presented because of repetition.

Characteristics to be Determined

The characteristics investigated were:

1. The number of grid points and DOF. This is an estimate of the size of a typical wing model.
2. Whether the model has leading and/or trailing edge secondary structure. Since the aero and structural models should both have the same planforms structural models without the secondary structure will require modification.
3. If the model has been validated with only the 2-DOF in the analysis set. The model can be validated either by known static deformations or by known natural frequencies. This should have

been done because the current NASTRAN aero theories allow only the Z-DOF in the analysis set (Ref 1).

4. The existence of any stresses, displacements, basic aerodynamic coefficients, and distributions for flight test data and standard computer analysis. This provides a measurement standard for judging the accuracy of results. Computed versions of these items can be generated by the solution sequence to provide accuracy checks at various points in the sequence. The data from a standard computer analysis could be used to check the sequence before it enters the iterative loop. The flight test data provides a check for the final results from the solution sequence.
5. Types of area elements on the wing surface. Currently, only certain element types on the wing surface can be analyzed by the solution sequence.
6. If there are any bending elements in the model. This will affect the validation described in question 3 since the analysis set for the solution sequence is condensed to only translational DOF by Guyan reduction.
7. If there are any SEQGP cards. This would affect the internal equation numbering which in turn affects the calculation of some of the transformation matrices (Ref 1).

8. If the wing model includes wheel wells or other cutouts. This will become significant when aero theories are implemented that panel 3-D structures. The current theory uses 2-D lifting surfaces and is only affected by the curvature of mean camber surface (Ref 1).
9. Is it in NASTRAN format. Considerable time and money can be spent in reformatting data to NASTRAN format.

Characteristics of Wings in Survey

Several wing models were evaluated with respect to the criteria. These models are T-37, T-38, B-1, C-5, and KC-135. Also, a wing model termed the "Eglin Wing" was included in the survey. The four wings presented in detail are the T-37 wing, the T-38 wing, the "Eglin Wing," and the B-1 wing.

The T-37 is illustrated here because it is a typical early subsonic wing type (Fig 1). It only has two spars and a thin skin. Unlike modern fighters, the T-37 wing has a high aspect ratio and a complex wing carry-through connected to the fuselage. This carry-through will present some problems in determining the wing root boundary conditions for this type of wing. The T-37 wing model has 319 grid points with 850 DOF. The model has no leading or trailing edges and, therefore, no control surfaces. This particular wing model has been validated in the Z-DOF only. This is unusual, typically wing models contain rotational



Fig 1. Graphical Description of T-37 Wing Model

DOF. The T-37 wing model is validated under such conditions due to the Guyan Reduction efforts being done at SAALC on the T-37 wing model for the FLEXLOADS program. There is very little flight test data because the T-37 was built when extensive tests on new aircraft were not done. There is some aerodynamic computer analysis for the T-37 wing, primarily by the airloads program called USSAERO. Several types of area elements are used on the wing surface. There are very few bending elements encountered in the T-37 wing model, they are only in the wheel well. The T-37 wing model contains SEQGP cards because the grid numbers are encoded position coordinates. Finally, the T-37 wing model is already in NASTRAN format.

The T-38 wing is a typical modern fighter-type wing having several spars, thick skin, a low aspect ratio and 388 grid points with 1564 DOF (Fig 2). It does have a leading edge structure but no trailing edge because the control surfaces are typically analyzed separately from the wing. The model has not been validated using only the Z-DOF. There are large amounts of flight test data and computer analyses to provide check data for the solution sequence. There are four types of area elements on the wing surface, some of which are plates, and there are additional bending elements within the wing. There are SEQGP cards included in the model. Wheel wells are present in the model but the wheel covers have not been modeled.

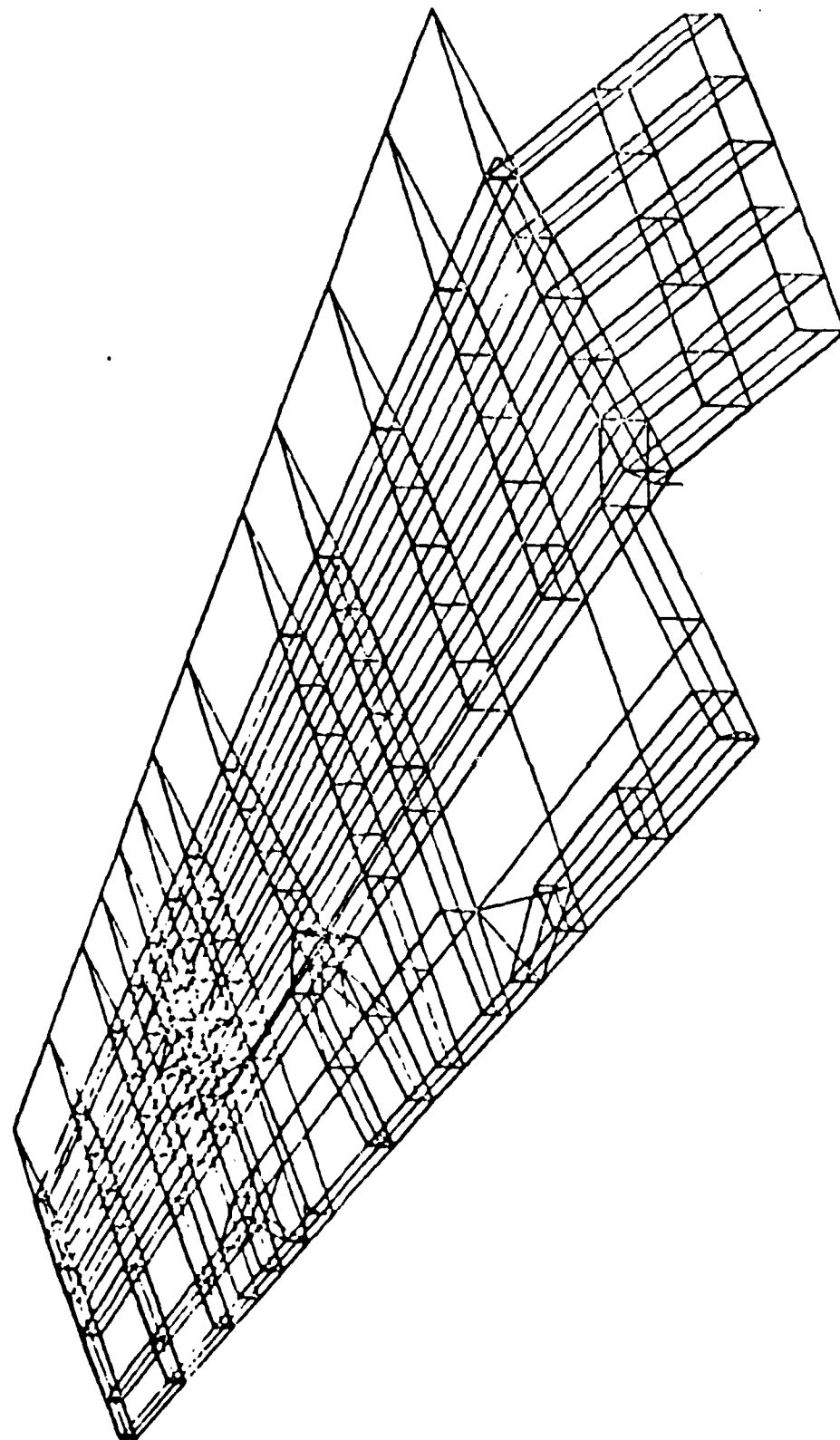


Fig 2. Graphical Description of T-38 Wing Model

Analyzing the wing is simplified by the fact that it is in NASTRAN format and has been analyzed in several static load conditions.

The Eglin Wing is presented because it is a typical damage tolerance wing model (Fig 3). These tend to be more complex because they have to closely trace the stresses throughout the wing. As a result, the Eglin Wing has 542 grid points with 2710 DOF. The model has both leading and trailing edges, the trailing edge consists partly of the control surfaces. It has not been validated with only the Z-DOF. The check data available is a minimum with only a small amount of computer analysis provided. There are several types of area elements on the wing surface in both triangular and quadrilateral shapes. The Eglin Wing does have bending elements. A large number of SEQGP cards are present which will affect the transformation matrices. The model also contains wheel well cutouts whose covers will have to be modeled if a 3-D aero theory is implemented. As with the T-38 wing model, the Eglin Wing is also in NASTRAN format and has been run in several static load cases.

Despite the effects of the wing pivot area, the B-1 wing model is taken as a typical "heavy-aircraft" type wing. These effects do not interfere with the present analysis so swing wing aircraft can also be analyzed. The outboard sections of the wing are fairly typical to other "heavy-aircraft" wings. Because the wing is large and complex, there are a great number of grid points and DOF, 5116 grid points with 24,361 DOF (Fig 4). Fortunately,

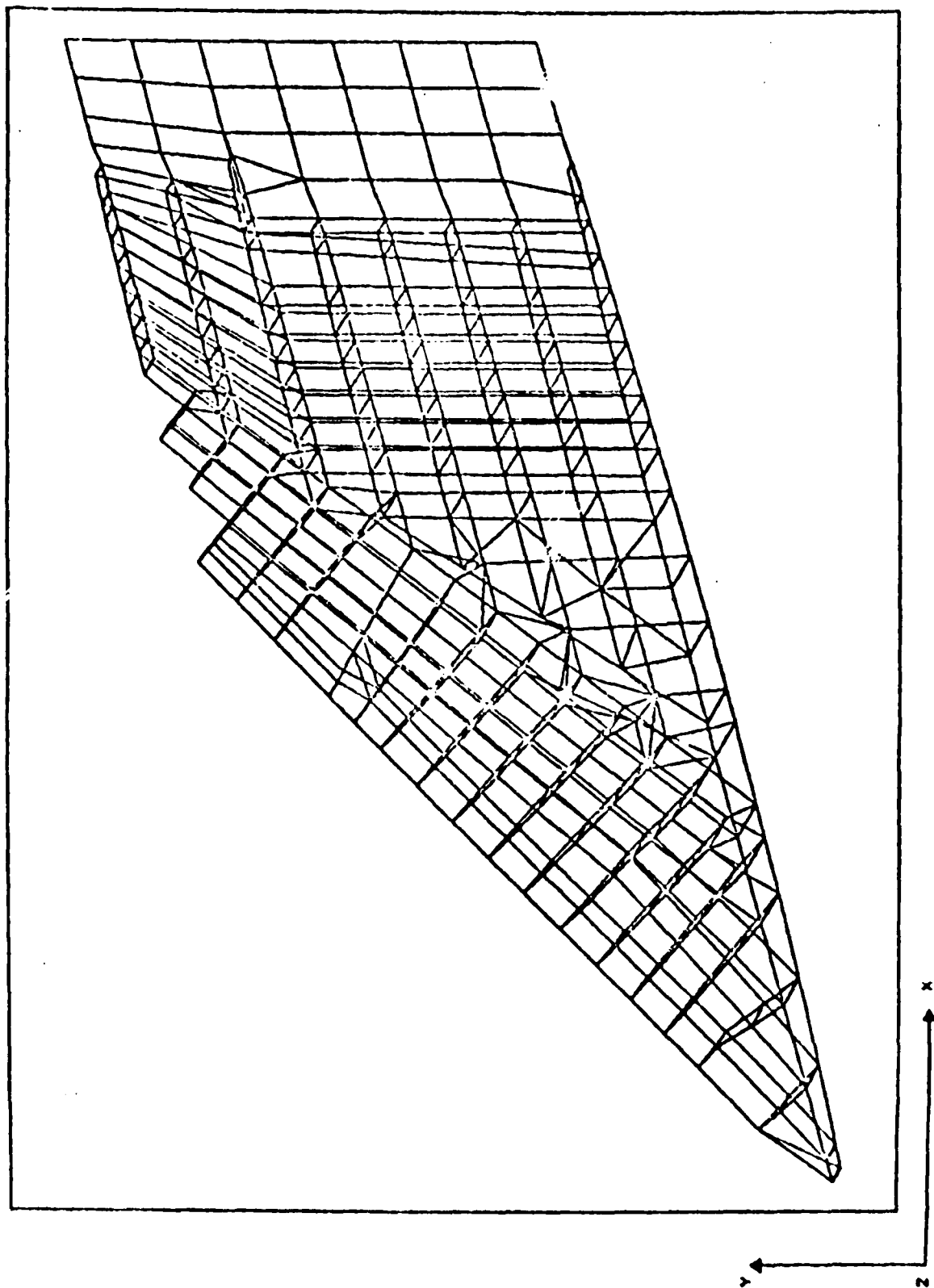


Fig 3a. Graphical Description of "Eglin Wing" Model

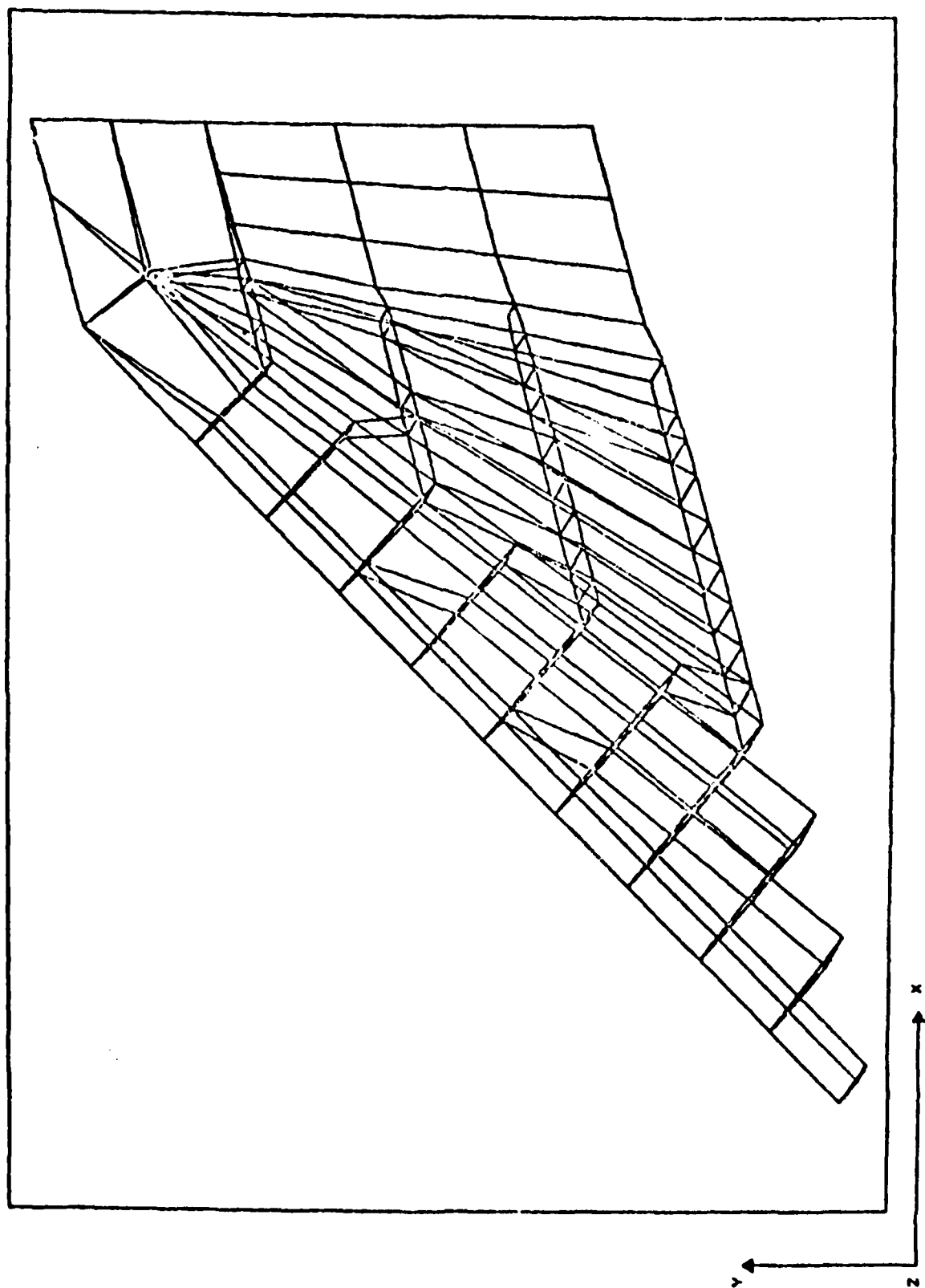


Fig 3b. Graphical Description of "Eglin Wing" Model (wing Tip)

INBOARD WING SUBSTRUCTURE

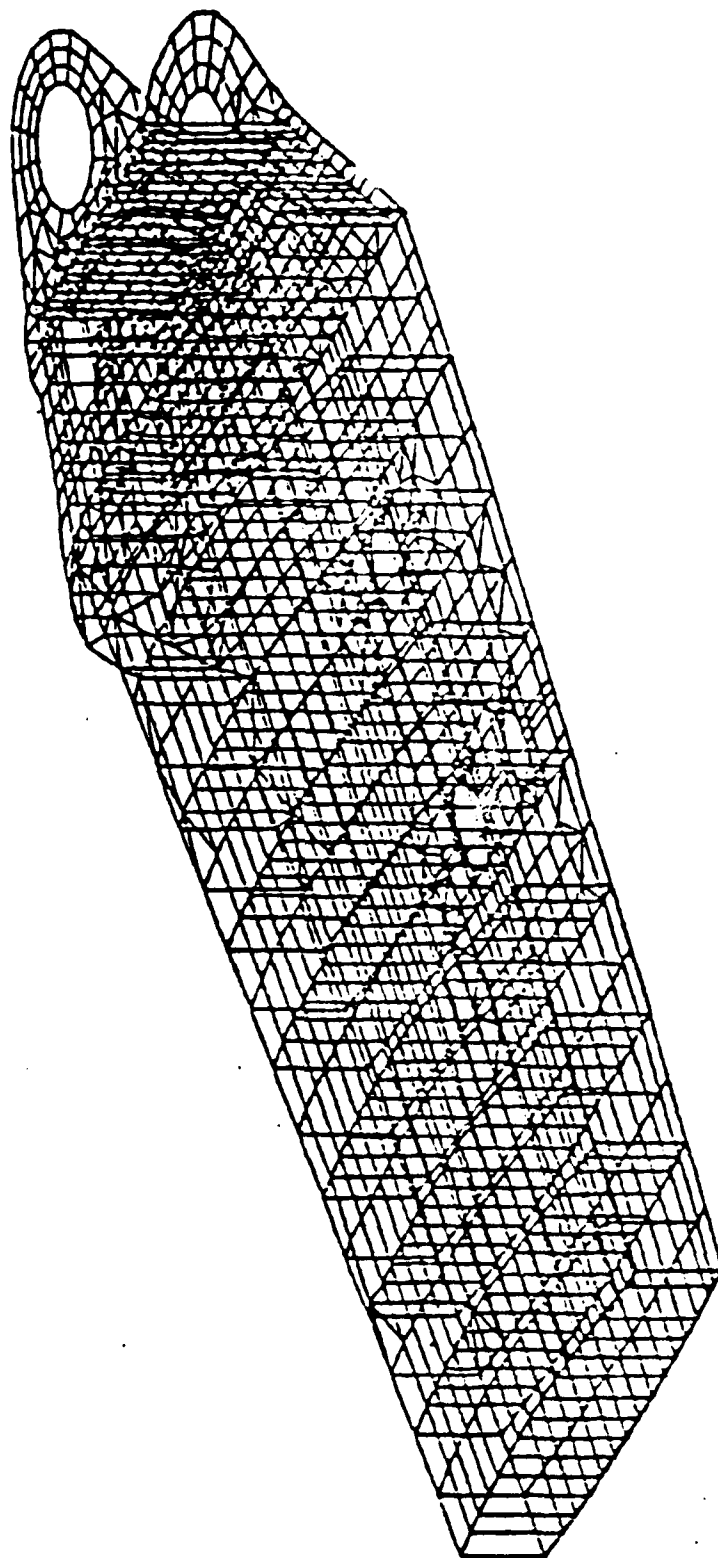


Fig 4a. Graphical Description of B-1 Wing Model (Inboard Substructure)

OUTBOARD WING SUBSTRUCTURE

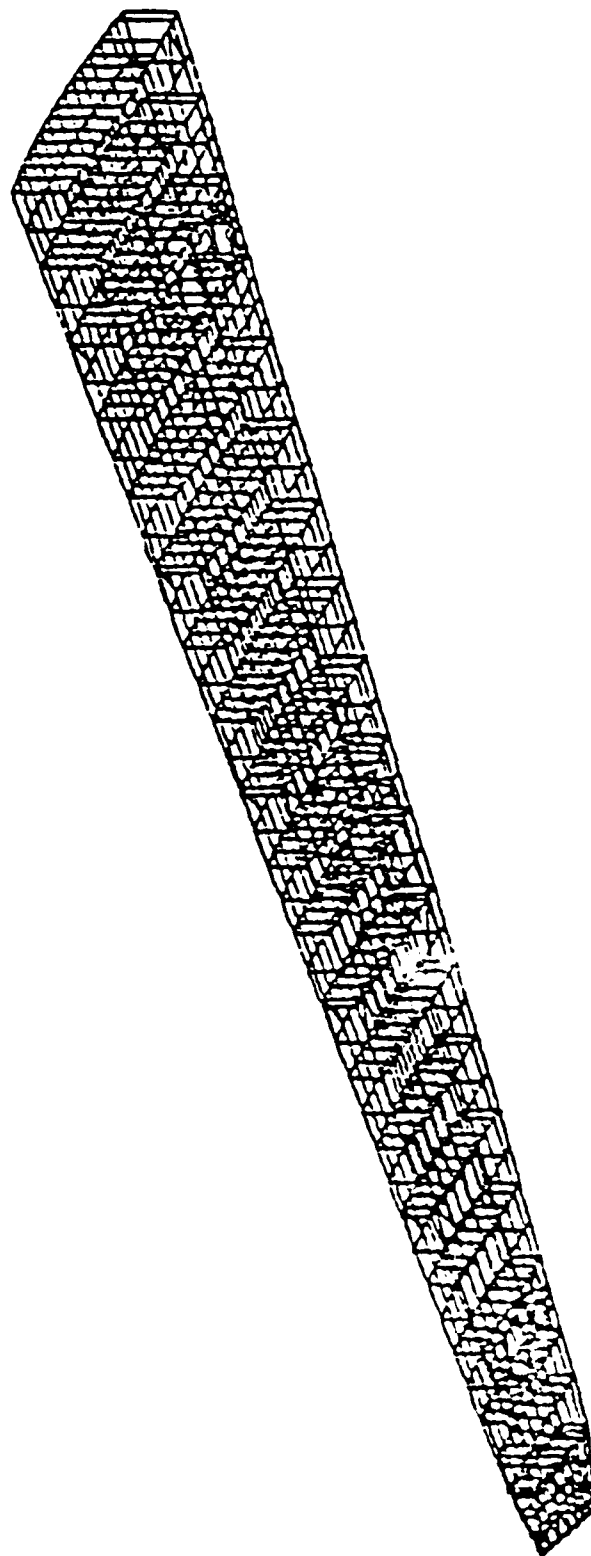


Fig 4b. Graphical Description of B-1 Wing Model (Outboard Substructure)

wings of this size can be analyzed using substructuring. The largest substructure in the B-1 wing model has 1605 grid points and 5479 DOF. This is still large but it is much more manageable than the entire wing. All parts of the wing are represented by the substructuring, including the leading and trailing edges, with the control surfaces. The B-1 wing model has not been validated using only the 2-DOF. There has been a lot of flight test data and computer analysis done on the B-1 wing. There are four types of area elements on the wing surface. The model includes bending elements, but no wheel wells or other cutouts. Since the B-1 model is not in NASTRAN format, SEQGP cards are not applicable.

Effects of Characteristics

Considering the wide range of characteristics of the wings in the survey, it is evident that compromises must be made. While the overriding concern in designing the solution sequence is to make it user-oriented, it is not possible to design the sequence for every characteristic. At times certain models will have to be altered to fit the sequence. The average size of wing models is fairly large and requires the sequence to use computer space efficiently. However, after the usage has been trimmed down as far as possible, there will still be some wing models that will be too large for the computer used. In these cases, either substructuring should be used or the complexity of the model lessened by Guyan Reduction

techniques.

If the structural wing model does not have leading and/or trailing edges, then two courses of action can be taken. One method is to alter the solution sequence to artificially transfer the loads from the leading and trailing edges to the structural wing box. However, this requires large amounts of the user's time. The second course of action would be to put leading edges on the wing model. This is the easier of the two in that, once done, it automates the load transfer from the edges to the wing box.

Since most wing models are not validated in the Z-DOF, there will be some error in using these models. The error can be measured by restricting the model to the Z-DOF, performing the validation procedures, and calculating how far off the model with only Z-DOF is from the actual structure. If the error determined is unacceptable, the model should be adjusted to give accurate values in the Z-DOF. Another course of action can be taken to reduce the error but it requires extensive rebuilding of the solution sequence. The only reason the models put through the solution sequence are reduced to Z-DOF is the aero analysis within the sequence cannot tolerate any displacements except out-of-plane translations. If desired, a different aero theory can be implemented into the solution sequence but it would require many months of effort to rebuild and debug the altered solution sequence.

Most wing models only use three or four area element

types on the wing surface. Given these element types, no problem is encountered with the solution sequence. However, if an element type different from those normally used is put on the wing surface then problems can arise. It is a simple matter to add that element type to the capabilities of the solution sequence, if sufficient knowledge of how the solution sequence works is available. Lacking that, it is advisable not to include the new element type in the wing surface.

Virtually every wing model contains bending elements. As with the validation of the models in the Z-DOF, the restriction of using bending elements in the solution sequence lies within the aero model. The bending elements will not be loaded by the aero forces in the rotational DOF. If the error incurred is unacceptable the recommended solutions are the same, either replace the bending elements or implement a different aero theory.

Most of the large wing models contain SEQGP cards in an effort to improve the bandwidth of the matrices involved. Currently there is no capability to tolerate SEQGP cards in the solution sequence so care should be taken in choosing the form of the input data for the wing model.

Most wing models contain wheel wells or other cutouts. This is not a problem with the present aero theory but it will need to be considered if a different aero theory is implemented. The most obvious solution is to include the wheel covers in the model. This will allow the wheel well

doors to be included as part of the lower wing surface.

A problem not previously mentioned is the aerodynamic interaction between the wing and other aircraft structures. If the wing is modeled alone then these effects are not taken into account. The aircraft structure that most commonly interacts with the wing is the fuselage. If the fuselage tends to be flat and vertical at the wing root, then the centerline symmetry condition of the wing alone will be a reasonable approximation. However, if the fuselage-wing root structure is rounded and smooth then there can be a great deal of crossflow. Other structures, such as engine inlets, engine exhaust, and wing mounted stores, can also affect the airflow across the wing. Wing-fuselage aerodynamic interaction can be modelled by constructing an aero model of the fuselage near the wing root.

The solution sequence can tolerate most of the characteristics of the average wing model. However, further revisions to the sequence would broaden its capabilities and applicability.

Bibliography

1. NASA SP-222(04). The NASTRAN User's Manual (Level 17.0). National Aeronautics and Space Administration, December 1979.
2. Ford, Douglas A., Advanced Development Projects, Lockheed Aircraft Corporation (Personal Interview). Lockheed, California, October 21, 1982.
3. Chrisinger, Lance E. Calculation of Airloads for a Flexible Wing Via NASTRAN. M.S. Thesis. DTIC AD A094770, Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December 1980.

STATIC AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART II

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

By

Kim Jones, B.S.A.E.

1Lt

USAF

Graduate Aerospace Engineering

December 1982

Approved for Public Release; Distribution Unlimited

Contents

<u>Part I.</u>	<u>Page</u>
I. Introduction.	1
II. Background.	2
III. Theory.	5
IV. Capabilities.	11
V. Procedure	13
VI. Survey.	14
 <u>Part II.</u>	
Abstract.	iii
I. Introduction.	1
II. Designing and Testing a NASTRAN Module.	2
III. Example of a New NASTRAN Module, GIAS	9
IV. Logical Procedures Within GIAS.	18
Output Datablocks.	18
Logical Procedures	19
V. Running the Flexible Wing Solution Sequence	22
Logical Procedures	22
Implementation	26
Bibliography.	31
Vita	32
Appendix A: GIAS and Environment	33
Appendix B: Sample Input Data and Illustration	79

Abstract

This thesis is a continuation of the research done by Captain Lance P. Chrisinger in his M.S. Thesis at AFIT in 1980. Captain Chrisinger developed a basic procedure to enable NASTRAN to analyze flexible wing airloads and stresses. That capability is expanded in this report to enable analysis of standard wing models.

A subroutine was incorporated into NASTRAN to eliminate extensive hand-calculation of transformation matrices.

The capability to tolerate control surfaces was discovered. Also, a survey of wing models in the Air Force inventory was taken to determine the characteristics of the average wing model. This was used to determine where the aeroelastic procedure was lacking in its ability to analyze wing models.

STATIC
AEROELASTIC ANALYSIS
OF
FLEXIBLE WINGS
VIA NASTRAN
PART II

I. Introduction

This part presents a detailed account of the procedures associated with the NASTRAN instruction sequence for flexible wings. However, the presentation presumes a level of knowledge about NASTRAN and its organization. In order to fully understand the majority of this part, the reader must understand the organization of NASTRAN, to include datablocks, modules, links, and overlays. The reader should have taken, or have notes of, a course dealing with NASTRAN System Programming (Refs 3,4).

This part is organized into three sections. The first two describe a recommended procedure for designing and testing a module to be put inside NASTRAN. This will also include, as an example, the module designed to work within the NASTRAN instruction sequence to solve flexible wing problems. These two sections were included to illustrate a set of steps not given elsewhere on designing and testing a NASTRAN module. The third section in this part is a detailed account of how to run this instruction sequence, with an explanation of additions to the Bulk Data Deck.

II. Designing and Testing a NASTRAN Module

The following is a recommended procedure for designing and testing new NASTRAN modules. The procedure carries through from the initial conception of the need for a new module to the final testing of the module once it is inside NASTRAN. The procedure pulls together information from various NASTRAN publications to present a complete, logical sequence of steps to design and implement a module. These steps are designed to insure the most error free module in the least amount of time. These will minimize the debugging required while manipulating the entire NASTRAN package. To do this the module must be tested in an environment as close to NASTRAN as possible before the module is entered into NASTRAN. The recommended steps in designing and testing a NASTRAN module are:

1. Recognition of the Need. In designing a new DMAP sequence or running an extensive DMAP alter the user might recognize that none of the DMAP statements, nor any combination of them, provide him with a procedure he needs. The first thing he must do before considering adding a module is be sure that there is no other way to accomplish the procedure, even by a round-about method. The time and effort required to

design, test, and implement a new NASTRAN module is enough to make it a last resort. If it is at all reasonable to operate without the module then that course of action is recommended. In considering a DMAP to solve his problem the user might compare MSC NASTRAN to COSMIC NASTRAN. The MSC version contains many more modules and might have the proper DMAP modules where the COSMIC version did not.

2. Development of the logic process and recognition of required inputs. These two steps are lumped together because the logic processes inside the module must operate within the NASTRAN environment. Some of the inputs that would be ideal for the procedure might just not be there, or might exist but in a format that will increase the complexity of the module's logic. To determine exactly what is needed by the sequence from the module, and also to check the sequence, run the sequence with the module and its outputs replaced by hand-generated data.

3. Determine where to get the inputs.

There are several ways that inputs are available to a module. They can either be: an existing datablock, a datablock that can be built in the

DMAP before the module is executed using DMAP alters or the data that can be user supplied through PARAM, DTI, or DMI cards. Datablocks can be either matrices or tables. The way the data reaches the module will have an effect on its logic processes and on the amount of user input needed. It is desirable to require as little as possible from the user. This will reduce input errors.

4. Write the module with all of its NASTRAN-necessary parts. NASTRAN uses the Fortran IV computer language. Any limitations with this language must be taken into account when writing the module. The module will be operating in an environment with common statements, open core arrangements, and various other subroutines to which it has access. Depending on the design of the module, the use of these will have to be carefully planned out so as to minimize the module's use of computer time and space. This must be considered because the large variation of problem types encountered will cause the solution sequence to tax the capabilities of most any computer.
5. Build a file of the data the module will receive. This will be the input to a simulated environment built to debug the module as thoroughly as

possible before it is put inside NASTRAN. The content of the data should be from several sample cases to which the correct (or at least expected) answers are known. The form of the data can be up to the user but it should contain all the information that will reach the module, not just what the module will use. The procedures to develop this file are determined by how the module will get the data when it is incorporated into NASTRAN. To get the data that the user will supply to the module when it is incorporated into NASTRAN is relatively simple. Just put the data on the input file as it would appear in the Bulk Data Deck. The other data, those that are generated inside NASTRAN before the module, will require a different approach. A solution sequence nearly identical to the one that will use the module should be run. This solution sequence should not have the call to the module being developed. It is not necessary for the sequence to run until completion. All that is required is to derive the data that the module will use. This data should be punched out of a NASTRAN run onto a file. It is advisable to do this right after the last DMAP statement before the new module is called. It is easier to do this with a DMAP alter. The specific cards that

need to be inserted in the DMAP depends on the form of the data. If the data is in table format use the TABPCH module. If it is in matrix format use MATPUN.

6. Build environment around the module to simulate the NASTRAN environment. Ideally, this should be done with as few changes to the module as possible. The form of the module should be as close as possible to the form it will have when inside NASTRAN. The environment should consist of several subroutines, including the module, with a master sequence of commands whose job it will be to call an initialization subroutine and then the module. The module, in turn, will call utility subroutines. The environment should simulate all the common statements, open core, functions, and subroutines that the module will see while inside NASTRAN. The environment subroutines need not do exactly what the equivalent NASTRAN subroutines do, but they must return the same response to the module. Some of the subroutines the module calls may have to be copied from NASTRAN source code and put into the environment rather than just building similar ones. However, caution should be used in doing this because the subroutines pulled from NASTRAN will

have to have their NASTRAN environments simulated also. Their environments can be as simple as another utility subroutine or they can require an assortment of common statements, open core, functions, and subroutines of their own.

7. Run several test cases to debug the module in the environment. The test cases should use the input data described in Step 5. It is helpful here to put internal check statements within the environment. Perhaps printing out a simple statement telling which subroutine the program is in. The module must be debugged until the environment's output data is exactly what the user wants NASTRAN to see once it finishes executing the module.

8. Put the module inside NASTRAN.

The procedure to do this can be found in the NASTRAN Programmer's Manual, in most any other NASTRAN system programming text or from someone who has done it before and is knowledgeable about the NASTRAN system. The module must be put in a link that will allow it access to the subroutines and common areas the module needs.

9. Run additional tests on installed module until satisfied. These should begin with the same test cases run before, in Step 7. However, it might be a good idea to run additional cases. It could be helpful to have various matrices and tables

printed out at key locations within the DMAP. Of course, to run these cases, the trial DMAP should have been altered to include the call for the module.

III. Example of a New NASTRAN Module, GIAS

This section uses the NASTRAN module GIAS as an example of the procedure to design a module.

GIAS was designed to build various matrices used to solve the flexible wing problem. Beforehand, the matrices were manually calculated and entered into the program via DMI cards. The overriding concept in GIAS is to minimize the work the user must do. As many matrices as possible are built within GIAS. This leaves only one matrix, other than the normal model data, to be built by hand. As development on GIAS continues, this matrix will also be built inside NASTRAN.

The following is a reiteration of the steps necessary to design a module for NASTRAN, with specific application to GIAS.

1. Recognition of need. The solution sequence for solving flexible wings requires several transformation matrices and moment arm vectors. The procedure used to generate these matrices is much too detailed and complex for NASTRAN to accomplish it with DMAP. Therefore, in order to significantly reduce the amount of data to be entered by the user a new module is needed to calculate the extra data.

2&3. Development of logic process, recognition of required inputs and where to get them. Since the object of the module is to reduce the user's work most of the input data to the module already exists in NASTRAN while it is running. At this time there is only one additional table that needs to be input by hand. Those that already exist in NASTRAN are:

SIL - Scalar Index List

BGPA - Basic Grid Point definition table,
Aerodynamics

GPLA - Grid Point List, Aerodynamics

ACPT - Aerodynamic Connection and Property
Table

ECTA - Element Connection Table, Aerodynamics

Descriptions of the existing datablocks can be found in the NASTRAN programmer's manual.

The user-supplied table is:

TOSEL - Top Surface Element List

TOSEL contains a list of area element numbers on the top surface of the wing. The datablock is needed because the loads generated by the aerodynamic model in the solution sequence are only applied to the nodes of the elements listed in TOSEL. TOSEL contains one record for each type of area element on the upper surface.

The records are arranged so that the element type

code numbers are in increasing order. The first entry in each record is the BCD name of that element type. The entries following that, within each record, are the element numbers of that element type on the upper surface in increasing order. TOSEL cannot be derived inside NASTRAN; currently there is no method available to isolate the upper wing surface elements in a general wing model.

4. Write the module with all of its NASTRAN-necessary parts. The entire module is listed in the back of this volume (Appendix A). There are various items that need to be included in the module to allow it to operate within NASTRAN. For GIAS the NASTRAN-necessary parts are:

COMMON statements

COMMON//A - Obtains the variable "A" out of the blank common. "A" is the bottom of blank common.

COMMON/SYSTEM/IBUF - IBUF is the required length of GINO buffers

COMMON/PACKX/TYPIN, TYPOUT, IROW, NROW, INCR - Various parameters for the PACK subroutine

COMMON/ZOPEN/Z(1) - Designates a common area for working

DATA statements

DATA NAME/4HGIAS,4HREAD/ - Assigns a BCD value for NAME, a parameter for the MESSAGE subroutine, an error utility

DATA statements are also used to assign inputs the consecutive numbers 101-106. In addition, data statements are used to assign outputs the consecutive numbers 201-209.

Functions

CORSZ(Z,A) - Determines amount of open core available from the field length specified for the program run

Subroutines

GOPEN - Opens a datablock in order to read from it or write into it

CLOSE - Closes a datablock

READ - Reads from a datablock

PACK - Writes to a datablock

RDTRL - Reads the trailer of a datablock

WRTRL - Writes the trailer of a datablock

MESSAGE - Prints selected fatal errors as they occur

SSPLIN - Produces an interpolation matrix
A more detailed description of the
subroutines can be found in the NASTRAN
Programmer's Manual (Ref 2).

5. Build a computer file of the input data the module will receive. A listing of the input data to the simulated environment for a test case is included in this volume. TABPCH cards were inserted in the solution sequence near line 104; this is where GIAS was later put. Datablocks ACPT, TOSEL, ECTA, GPLA, BGPA, and SIL were output to logical file PUNCH. This file was then manipulated, along with the initialization subroutine (see subroutine INIT, Appendix A), to put these datablocks in a form where the rest of the simulated environment can use them.
6. Simulate the NASTRAN environment. All of the items discussed in Step 4 must be duplicated within the simulated environment. In the simulated environment they do not have to do exactly what the subroutines in NASTRAN do, but they must return the same response to the module being tested. The following are the simulated items in the environment for GIAS; they have the same names as the items in Step 4.

Function CORSZ(Z,A) - Returns a predetermined number to be the size of the Z array.

COMMON/SYSTEM/IBUF - Returns a predetermined arbitrary number to be the size of the buffers, the buffers used in the simulation.

Subroutines

INIT - The initialization subroutine, reads the input file into internal arrays for use by the subroutines READ and RDTRL.

GOPEN - Prints out the statement "Datablock ### opened" to announce the position of the program, but it really does nothing to the datablock.

CLOSE - Prints out the statement "Datablock ### closed" to announce the position of the program, but it really does nothing to the datablock.

READ - Fetches all or part of the datablock contained within a set of arrays prepared by subroutine INIT. It is more complex than an average read statement so it can handle any

errors it encounters, such as being asked to read past an End-of-File or an End-of-Record.

PACK - Prints out the name of the specified datablock along with its values, for inspection but does not create any files.

RDTRL - Reads the trailer of the datablock specified from the set of arrays prepared by INIT.

WRTRL - Prints out the trailer of the datablock specified.

MESSAGE - Prints out the specified error message.

SSPLIN - Produces an interpolation matrix. SSPLIN calls two other utility subroutines, INVERS and GMMATS. These must also be included in the environment. All three of these subroutines had to be copied directly from NASTRAN source code.

7. Run several test cases to debug the module. A simplified wing box was designed and run in NASTRAN using input entered by hand to replace

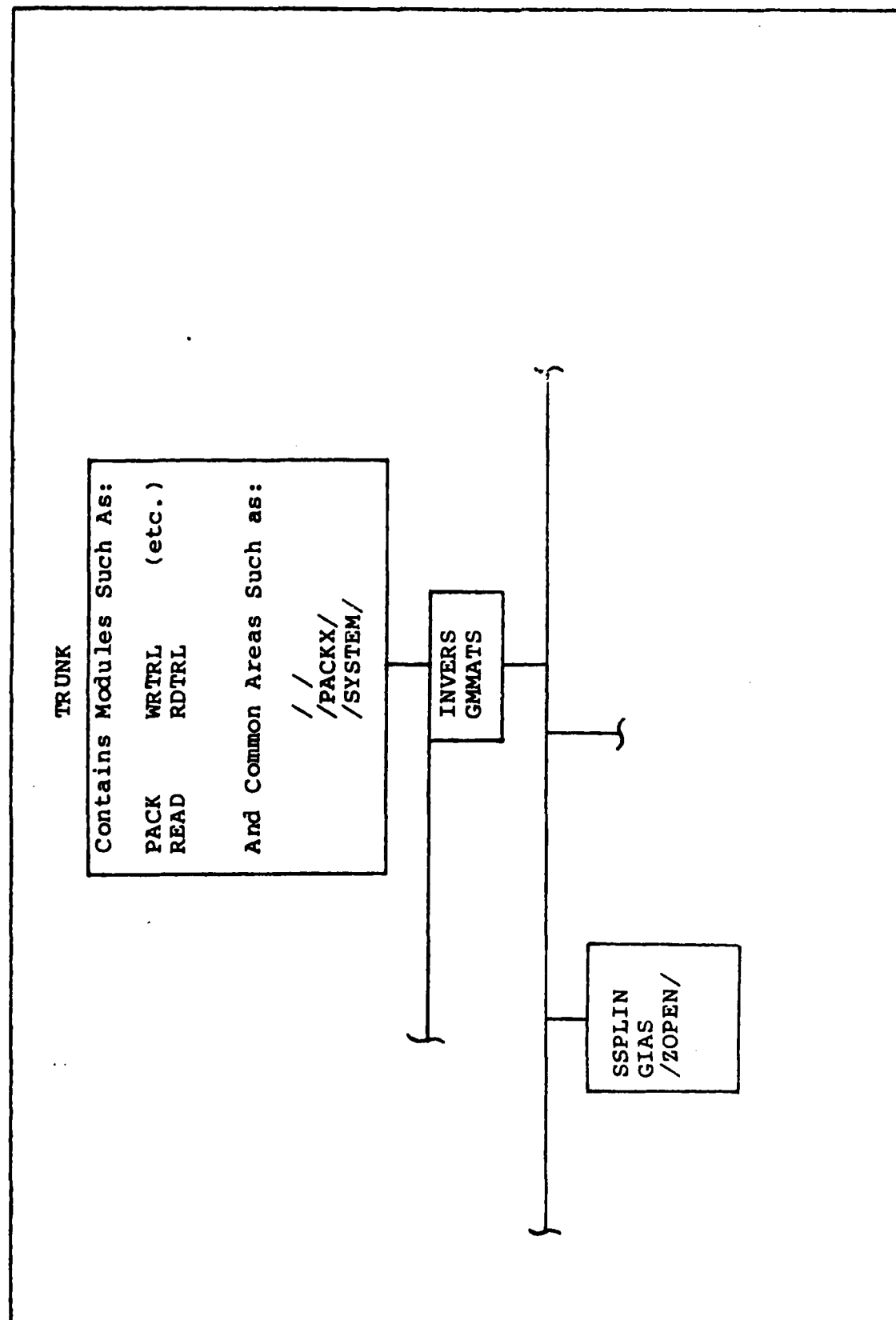


Fig 1. Link 11

GIAS. Running several variations of this produced the required check data for the environment.

8. Put the module inside NASTRAN. Because SSPLIN is one of the utility subroutines used, GIAS was put into Link 11 of NASTRAN (Fig 2). The other subroutines needed are also available from this position.
9. Run additional test cases until satisfied. Both the simplified wing box mentioned in Step 7 and an intermediate complexity wing were run as final test cases.

IV. Logical Procedures Within GIAS

It was previously mentioned that GIAS created various transformation matrices and moment arms. This section will look at each of these datablocks and outline the logical procedures used to make them. Several of the procedures overlap within GIAS because several of the output datablocks use the same input.

Output Datablocks

The output datablocks are:

- SAS - A matrix containing ones where areas are in the SKJ matrix
- GTAK - Surface spline interpolation matrix for transformation of ΔP 's at the aero points to ΔP 's at the structural nodes
- TTTTT - Transformation matrix from ΔP 's at the structural points to loads at the structural nodes
- VGLS - A vector with ones in the proper positions so that when multiplied by the structural loads it sums them. VGLS also condenses PG down to nonzero terms.
- VTMA - Contains the moment arms for the aerodynamic pitching moment about the Y-axis.

- VTRA - Contains the moment arms for the aerodynamic rolling moment about the X-axis
- VTM - Contains the moment arms for the pitching moment about the Y-axis due to the transformed structural loads.
- VTR - Contains the moment arms for the rolling moment about the X-axis due to the transformed structural loads.
- AIDMT - Unit vector of length equal to the number of aerodynamic boxes.

Logical Procedures

The first datablocks built in GIAS are TTTT and VGLS. Even though they are built at the same time the logic to build each of them will be considered separately. TTTT transforms the Δp applied to the centroid of each area element on the upper surface of the wing model to loads on the upper surface structural nodes. The first step to do this is to multiply the Δp applied to the element by the area of the element. Using the geometry of the element the load over the element is then correctly partitioned out to each node of the element. The only information needed to calculate TTTT is the coordinates of the structural nodes on the upper surface and which elements they are associated with. Since VGLS is a list of ones in the Z DOF position for each node on the upper surface the only information needed is the upper surface node numbers.

The next datablocks built are VTM and VTR. The moment arms for the structural rolling and pitching moments are simply the coordinates of each upper surface node. This information was obtained when TTTT and VGLS were built.

VTMA and VTRA are built next. The only information needed for these two is the coordinates of the corners of the aero boxes. From these the coordinates of the 1/2-span 1/4-chord point of each aero box is found. This is the point of each box where the lift is assumed to act. The X and Y values of each of these are the entries in VTMA and VTRA, respectively.

All the information needed to calculate GTAK is now available. In order to calculate GTAK the subroutine SSPLIN needs to know the coordinates of the input and output points and how many of each there are. The input points are the points on the aero boxes where the pressure is assumed to act. The output points are the upper surface structural nodes. Because GTAK has to be built all at once, rather than a column at a time like other matrices, this is the procedure in GIAS that takes up the most computer memory. In fact, depending on the problem, this procedure could be the one that governs the amount of computer memory required for the entire analysis.

The last two datablocks built by GIAS are AIDMT and SAS. To build AIDMT only the number of aero boxes is

needed. SAS has ones down two of its diagonals and zeroes elsewhere. It has as many rows as two times the number of aero boxes and as many columns as the number of aero boxes.

V. Running the Flexible Wing Solution Sequence

In order to best use a solution sequence it is advisable to first understand how it works. Therefore, this section first presents a detailed outline of how the solution sequence for flexible wings operates. Afterward, an explanation of the Bulk Data Deck is presented. An example of the input deck is given in Appendix B.

Logical Procedures

The first part of the solution sequence (from now on, called DMAP) sets up the tables that describe the structural model. These include ECT (Element Connection Table) and GPL (Grid Point List). This set of matrices describes the unconstrained structural model. The constraints of the first subcase allow only rigid body pitch about the wing root trailing edge. This will be found in the eigenvalue analysis and used to set the initial angle of attack. The frequency limits of 0.0 to 1.0 on the EIGR card ensure that this is the only eigenvector used. The DOF are then further partitioned down by Guyan Reduction to only the remaining Z DOF since the aero model can only tolerate displacements in the Z-direction. A real eigenvalue analysis is then performed within a small frequency range around zero. The purpose of the analysis is to obtain the mode shape of the rigid body

pitch mode. The Z-displacement of the most in-board leading edge node, not including the wing root, is then normalized to one by selection of "point" on the EIGR card. All displacements are multiplied by a user calculated parameter, AOAP, in order to set the initial angle of attack (Fig 1). This procedure determines the zero-camber initial displacements of all the nodes. The indirect method of determining this is unavoidable because COSMIC NASTRAN has no module, or simple sequence of modules, that will do this. Camber is added to the wing, if desired, by using a parameter ICAMB and a matrix CAMBER. After this, various transformation matrices are built. The first one created is GTKA, which is built using the Bulk Data SPLINE cards. This will be used to transform the displacements of the structural model to values of slope on the aero model. These values will then be used by the aero analysis portion to specify the strengths of the downwash vector, DIJK or D2JK. This vector is then multiplied by the aero influence matrix for the Doublet-Lattice aero method, AJL^{-1} to determine the ΔC_p distribution on the aero model. Following the calculation of GTKA the module GIAS is called to determine other transformation matrices and moment arms. This ends the first subcase of the DMAP.

The second subcase is distinguished from the first by a change of SPC cards. The structural model is reconstrained to allow no rigid body motion. None of the nodes on the wing root have any active DOF; all the other

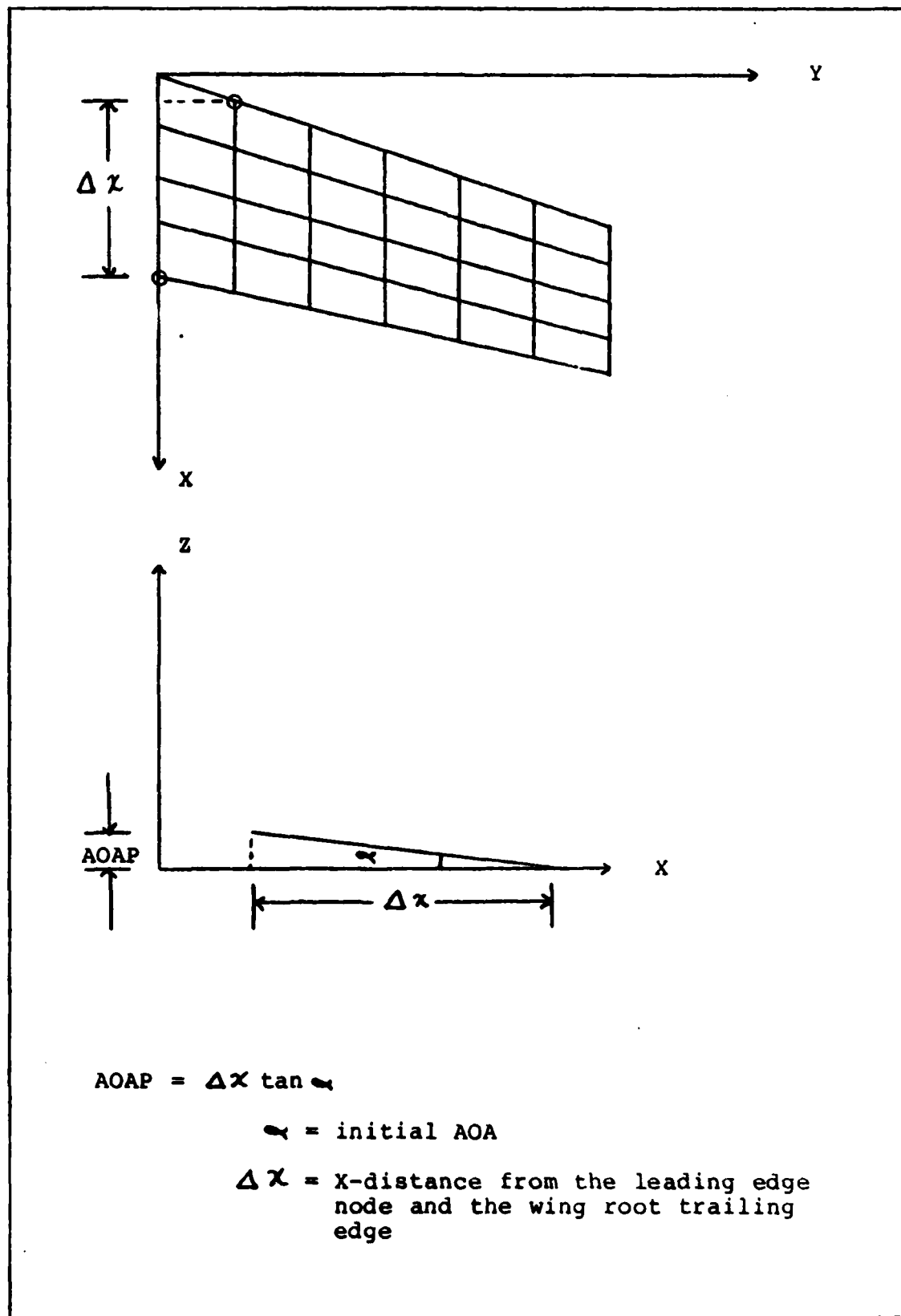


Fig 2. Determining AOAP

nodes are limited to only translational DOF. These nodes are then further reduced to only the Z DOF by ASET cards. This procedure locks the wing root at the initial AOA as determined by the eigenvalue analysis and AOAP. Then the matrices necessary for static deflection analysis are determined for this configuration. This analysis is then performed to determine the structural loads due to the initial AOA.

The next section is the most significant one of the DMAP. The iterative loop calculates the displacements, structural loads, and aero forces due to the structural deformations. Then the structural deformation due to the new airloads are calculated. The loop begins by finding the change in structural deformations due to the airload vector. On the first iteration the load vector is due to the initial AOA. On subsequent iterations it is due to the change in structural deformations. Then the incremental deformations are transformed to slopes on the aero model using the GTAK generated from the Bulk Data SPLINE cards. These slopes are used to determine incremental ΔC_p . The incremental ΔC_p are transformed to an incremental load vector which is used in the next iterative pass through the loop. This is accomplished by premultiplying the ΔC_p vector by GTAK and then by TTTTT. At the end of each increment the changes in deformations, structural loads, and aero forces are accumulated. The loop is finally left when the norm of incremental displacements (VNSR) is less

than the user supplied parameter ELOOP, or a total of ten incremental passes have been made. The loop gives the final values for the structural deformations, structural forces, and ΔC_p vector.

The DMAP then recovers the complete solution for both the analysis set and the omitted DOF. A complete stress recovery for all the structural elements can be done if requested by the user. The final section of the DMAP determines C_L , C_{np} , C_{nr} due to both the final aero forces and the final structural loads. Both sets are calculated to provide a means of checking the accuracy of the force transformations between the aero and structural models.

Implementation

In addition to the usual model used in structural analysis there are a few more cards, and one additional datablock, needed to run the DMAP. The extra datablock is TOSEL and is entered via DTI cards. TOSEL is the table of area element numbers on the upper wing surface. The records in TOSEL are organized according to element BCD names. A more detailed explanation of TOSEL is presented in Section III of this volume. The additional cards needed are (Ref 1):

- ASET1 - Defines the DOF in the analysis set. For the DMAP the analysis set should consist of only the Z-DOF for every non-wing-root node.
- CAERO1 - Divides the aero panels into equal boxes for the Doublet-Lattice aero theory. The param-

eters on this card are determined by the wing planform and the user's needs.

- AEFACT - Divides the aero panels into unequal boxes, used in conjunction with the CAERO1 card.
- PAERO1 - Defines associated bodies for the CAERO1 card.
- EIGR - Lists parameters for the real eigenvalue analysis. The frequency range of interest should be from 0.0 to 1.0 in order to include only the rigid body pitch mode. There is only one estimated and desired root in the frequency range. The eigenvector should be point normalized on the Z-displacement of the most in-board leading edge node, not including the wing root.
- AERO - Defines the basic aero parameters. The DMAP uses the parameter Q, the dynamic pressure, and the mach number from the MKAERO card instead of the parameters on the AERO card. Therefore the values on the aero card do not matter, but they have to be positive to allow the DMAP to complete the processing of the AERO card.
- MKAERO - Defines the mach number and reduced frequency for the desired flight condition. In the DMAP only one mach number and reduced frequency are allowed. The mach

number should be subsonic and representative of the desired flight condition. The reduced frequency should be a positive number very close to zero.

SPLINE1 - Defines the parameters for generation of a surface spline to interpolate the out-of-plane displacements on the structural model to out-of-plane displacements and slopes on the aero model. The **SPLINE1** card is used in exactly the same way as it is in the flutter analysis rigid format.

SET1 - List of structural grid points to be used with the **SPLINE1** card. Normally the list is only the nodes on the upper surface.

PARAM cards

AOAP - Used to set initial AOA. Real number. It multiplies the rigid body pitch mode eigenvector after the point normalization is done as specified on the **EIGR** card.

ELOOP - Tolerance of convergence for the iterative loop. It is compared to the norm of the change of deformations for each iteration. Real number.

LMODES - Number of modes to be used in the modal flutter formulation. Since

the only desired mode is the rigid body pitch mode LMODES is one.

LMODES is normally used when several eigenvectors are considered in a flutter analysis. In these cases the different eigenvectors are arranged as columns in a forcing vector for determination of modal pressures.

LMODES would then be the number of columns. LMODE is a real number.

- Q - Dynamic pressure. Must correspond to the desired flight condition.
Real number.
- RC - Root chord length. Used in formulating aerodynamic coefficients.
Real number.
- WAREA - Planform area of the model. Used in calculating aerodynamic coefficients. Real number.

As an option, camber can be added to the wing. If camber is required then an additional PARAM card should set ICAMB = 1. Also, the CAMBER vector must be input via DMI cards. CAMBER is as long as the number of corners on the aero boxes. If two or more aero panels are used then the common nodes on both panels are counted twice. The values

that are in CAMBER are the Z displacement of each aero box corner due to the camber of the airfoil. In addition, any control surface deflection can be treated as if it were a change in camber.

Finally, in order to run this DMAP for aeroelastic analysis of flexible wings, the case control and executive card decks must be formulated. The executive card deck must reflect the fact that a DMAP is being run rather than a rigid format. The case control deck notes the set identification number of the EIGR card in the Bulk Data Deck. The case control lists the two subcases and the SPC identification numbers for each. In addition, output requests as needed are specified in the case control. The values of the first subcase correspond to the initial pitch configuration; those of the second subcase are a final result of the iteration loop. All the available output requests for the static loads analysis rigid format are also available for the DMAP.

Bibliography

1. NASA SP-222(04). The NASTRAN User's Manual (Level 17.0). National Aeronautics and Space Administration, December 1979.
2. NASA SP-223(04). The NASTRAN Programmer's Manual (Level 17.0). National Aeronautics and Space Administration, December 1979.
3. Hurwitz, Myles H. Course Notes of NASTRAN System Programming. MSN Software Services, 1978.
4. Schaeffer, Harry G. Course Notes of Direct Matrix Abstraction Programming (DMAP) in MSC/NASTRAN. Schaeffer Analysis, Inc., 1981.

Vita

Kim Jones was born on 28 February 1958 in Weisbaden, Germany. He graduated from high school in Clearwater, Florida in 1976 and was subsequently accepted into the Air Force Academy. A year after receiving a Bachelor of Science Degree in Aeronautical Engineering, he entered the School of Engineering, Air Force Institute of Technology, in June 1981.

Permanent Address: 837 Island Way
Clearwater, Florida 33515

Appendix A

LOAD MAP - CIASCHV

PLA OF THE LOAD 111
LWA+1 OF THE LOAD 33330

TRANSFER ADDRESS -- CIASCHK 15310

PROGRAM ENTRY POINTS -- CIASCHK 15310

PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK	ADDRESS	LENGTH	FILE	DATE	PROCESSR VER LEVEL	HARDWARE	COMMENTS
/ZOPEN/	111	4704					
/ZSIZE/	5015	1					
CIASCHK	5016	10301	LGO	11/12/82 PTM	4.8 564	666X I	PROGRAM OPT-1
/SYSTEM/	15317	1					
/PACKX/	15320	5					
CIAS	15325	1725	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
/ZDATA/	17252	1024					
/DTRAIL/	20276	44					
INIT	20342	1522	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
MESSAGE	22064	50	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
SSPLIN	22134	704	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
CHMATS	23040	215	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
INVERS	23255	340	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
CORSZ	23615	11	LGO	11/12/82 PTM	4.8 564	666X I	FUNCTION OPT-1
COPIEN	23626	20	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
CLOSF	23646	20	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
PACK	23666	114	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
PDTRL	24002	73	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
K: -TFL	24075	41	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1
READ	24136	310	LGO	11/12/82 PTM	4.8 564	666X I	SUBROUTINEOPT-1

THE REST OF THE LOADER MAP IS NONAPPLICABLE

```

1      PROGRAM CIASCHE 74/74 OPT=1
      PROGRAM CIASCHE(INPUT,OUTPUT,TAPE5,TAPE6)
      COMMON/ZOPEN/7(2500)
      COMMON/FSIZE/512
      SIZE=2500
      : CALL INIT
      : CALL CIAS
      STOP
      END
    
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
10272 CIASCHE

VARIABLES	SN	TYPE	RELOCATION	ZSIZE	O	Z	REAL	ARRAY	ZOPEN
FILE NAMES		MODE	2054	OUTPUT			4130	TAPE5	6204
EXTERNALS		TYPE	ARCS				INIT		
CIAS			0						

COMMON BLOCKS

ZOPEN	LENGTH
2500	1

STATISTICS

PROGRAM LENGTH	2648	180
BUFFER LENGTH	10015R	4109
CM LARFED COMMON LENGTH	4705R	2501

52000R CM USED

```

1      CCC
2      CCC
3      CCC
4      CCC
5      CCC
6      CCC
7      CCC
8      CCC
9      CCC
10     CCC
11     CCC
12     CCC
13     CCC
14     CCC
15     CCC
16     CCC
17     CCC
18     CCC
19     CCC
20     CCC
21     CCC
22     CCC
23     CCC
24     CCC
25     CCC
26     CCC
27     CCC
28     CCC
29     CCC
30     CCC
31     CCC
32     CCC
33     CCC
34     CCC
35     CCC
36     CCC
37     CCC
38     CCC
39     CCC
40     CCC
41     CCC
42     CCC
43     CCC
44     CCC
45     CCC
46     CCC
47     CCC
48     CCC
49     CCC
50     CCC
51     CCC
52     CCC
53     CCC
54     CCC
55     CCC
56     CCC
57     CCC
58     CCC
59     CCC
60     CCC
61     CCC
62     CCC
63     CCC
64     CCC
65     CCC
66     CCC
67     CCC
68     CCC
69     CCC
70     CCC
71     CCC
72     CCC
73     CCC
74     CCC
75     CCC
76     CCC
77     CCC
78     CCC
79     CCC
80     CCC
81     CCC
82     CCC
83     CCC
84     CCC
85     CCC
86     CCC
87     CCC
88     CCC
89     CCC
90     CCC
91     CCC
92     CCC
93     CCC
94     CCC
95     CCC
96     CCC
97     CCC
98     CCC
99     CCC

```

SUBROUTINE CIAS
CIAS ACPT,TOSEL,ECTA,GPLA,BCPA,SIL/SAS,CTAK,TTTTT,
VCLS,VTHA,VTRA,VTH,VTR,AIDMT/S
INTEGER SAS,CTAK,TTTTT,VCLS,VTHA,VTH,VTRA,VTRA,AIDMT
INTEGER ACPT,TOSEL,ECTA,GPLA,BCPA,BUF1,BUF2,BUF3,CORSZ
INTEGER TRAIL,MCBA,MCBN,TYPEIN,TYPEOUT,SIL,IZ
REAL Z
DIMENSION TRAIL(7),MCBA(7),MCBN(7),NAME(2),IZ(1)
EQUIVALENCE (IZ(1),Z(1))
COMMON //A
COMMON /SYSTEM/IRUP
COMMON /PACKX/TYPEIN,TYPEOUT,IRUP,NROW,INCR
COMMON /ZOPEN/Z(1)
STRING FOR MESSAGE UTILITY
DATA NAMEZ/ANGIAS,AREAD/
INPUT DATA BLOCKS LOCAL NAMES
DATA ACPT,TOSEL,ECTA,GPLA,BCPA,SIL/101,102,103,104,105,106/
OUTPUT DATA BLOCKS LOCAL NAMES
DATA SAS,CTAK,TTTTT,VCLS,VTHA,VTRA,VTH,VTR,AIDMT/
*201,202,203,204,205,206,207,208,209/
INCR=1
GET END OF OPEN CORE
LCORE=CORSZ(Z,A)
SET UP A BUFFER FOR USE WITH COPEN AND READ
BUF1=LCORE-IRUP+1
IF (BUF1.LE.0) GO TO 500
AN "N" IS BEFORE THE DATA BLOCK NAME SPECIFIES THE FIRST
ADDRESS OF THE BLOCK IN OPEN CORE
AN "L" BEFORE THE DATA BLOCK NAME SPECIFIES A PARAMETER
WHICH IS THE LENGTH OF THE DATA BLOCK IN OPEN CORE
DATA BLOCK ACPT WILL BE FIRST IN OPEN CORE
MACPT=1
OPEN 'AERODYNAMIC CONNECTION AND PROPERTY TABLE'
CALL COPEN(ACPT,Z(BUF1),0)
GET 5+2*NP WORDS OF ACPT
WORD 1 IF: 1 - DLM (DOUBLET-LATTICE METHOD)
2 - STRIP THEORY
3 -
4 -
5 -
6 -
7 -
8 -
9 -
10 -
11 -
12 -
13 -
14 -
15 -
16 -
17 -
18 -
19 -
20 -
21 -
22 -
23 -
24 -
25 -
26 -
27 -
28 -
29 -
30 -
31 -
32 -
33 -
34 -
35 -
36 -
37 -
38 -
39 -
40 -
41 -
42 -
43 -
44 -
45 -
46 -
47 -
48 -
49 -
50 -
51 -
52 -
53 -
54 -
55 -
56 -
57 -
58 -
59 -
60 -
61 -
62 -
63 -
64 -
65 -
66 -
67 -
68 -
69 -
70 -
71 -
72 -
73 -
74 -
75 -
76 -
77 -
78 -
79 -
80 -
81 -
82 -
83 -
84 -
85 -
86 -
87 -
88 -
89 -
90 -
91 -
92 -
93 -
94 -
95 -
96 -
97 -
98 -
99 -

11/03/82 20.07.46

FTN 4.F+544

74/74 CPT-1

SUBROUTINE CIAS

```

60      CALL READ(ACPT,Z(MACPT),3*2*NP,1,M)
      NI IS THE NUMBER OF AERO BOXES AND 1/4 CHORD, 1/2 SPAN AERO POINTS
      NI=12/(MACPT+1)
      CCC
      CLOSE ACCESS TO ACPT, BUT KEEP WHAT'S IN OPEN CORE ALREADY
      CALL CLOSE(ACPT,1)
      CCC
      FETCH TRAILER TO TOSEL (TOP SURFACE ELEMENT LIST) USER SUPPLIED
      ON DTI CARDS
      CCC
      TRAIL(1)=TOSEL
      CALL RDTRL(TRAIL)
      TRAILER OF TOSEL CONTAINS:
      CCC
      WORD 1 CINO DB NAME
      WORD 2 # OF ENTRIES IN TOSEL
      CCC
      WORD 3 0
      WORD 4 0
      WORD 5 0
      WORD 6 0
      CCC
      ND IS THE NUMBER OF STRUCTURAL ELEMENTS LOADED AND ALSO
      THE NUMBER OF DEPENDENT STRUCTURAL POINTS WHEN UNIFORM PRESSURE IS
      ASSUMED OVER STRUCTURAL BOXES
      CCC
      LATER ON (AFTER BUGS ARE ironed OUT) WILL READ TOSEL UNTIL #MEL
      THEN SAY ND=M
      CCC
      IT'S ALSO THE SIZE OF TOSEL
      ND=TRAIL(2)
      NTOSL=MACPT+3*2*NP
      READS IN ALL OF TOSEL
      REUSES 1ST BUFFER
      CCC
      CALL COPEN(TOSEL,Z(BUFI),0)
      CALL READ (TOSEL,Z(MTOSEL),ND,1,M)
      CALL CLOSE(TOSEL,1)
      GETS SIZE OF CPLA
      CCC
      CPLA (GRID POINT LIST-AERODYNAMIC)
      TRAIL(1)=CPLA
      CALL RDTRL(TRAIL)
      WORD 2 = # OF GRID POINTS + # OF SCALAR POINT (ASSUMED=0, + # OF
      EXTRA POINTS (ASSUMED=0) + # OF AERO POINTS
      CCC
      LCPLA=TRAIL(2)
      LSCPLA=4*LCPLA BECAUSE THERE ARE 4 WORDS PER POINT (WORD 1 = 1D#,
      WORDS 2,3,4 = X,Y,Z)
      CCC
      NGPA = BASIC GRID POINT DEFINITION TABLE - AERODYNAMIC
      INCPA=4*LCPLA
      READ IN ALL OF CPLA TO GET 1D#'S OF POINTS
      CALL COPEN(CPLA,Z(BUFI),0)
      NGPLA=NTOSL+ND
      CALL READ(CPLA,Z(NGCPLA),LCPLA,1,M)
      CALL CLOSE(CPLA,1)
      NGCPA=NGPLA+LCPLA
      NCORF=BUFI-1
      CHECK TO SEE IF ENOUGH OPEN CORE IS AVAILABLE
      CCC
      IF(NCORE.LT.(NGCPA+LSCPLA)) GO TO 506
      READ IN ALL OF NGPA TO GET COORDINATES OF POINTS
      CALL COPEN(NGPA,Z(BUFI),0)
      CALL READ(NGPA,Z(NGNGPA),LSCPLA,1,M)
      CALL CLOSE(NGPA,1)
      ICQWEN=5304 IS THE INTERNAL IDENTIFIER OF CONWEN2 ELEMENTS
      CCC

```

11/03/82 20.07.46

FTN 4.R+564

74/74 OPT-1

SUBROUTINE GIAS

```

115 CCC THE FOLLOWING 6 STATEMENTS LOCATE THE RECORD FOR THESE
CCC ELEMENTS IN THE ECTA TABLE
CCC OPENS ECTA FOR READING
CCC ECTA = ELEMENTY CONNECTION TABLE - APRODYNAMIC
120 CCC CALL COPEN(ECTA,Z(BUFI),0)
CCC FROM HERE UNTIL STATEMENT LABEL 2 NON-CQDNEM2
CCC ELEMENTS ARE SCREENED OUT, THE RECORD CONTAINING ALL
CCC THE CQDNEM2 ELEMENTS IS LOCATED AND POSITIONED FOR READING
ICQNM=5308
NECTA=NBGPA+LNGPA
125 1 CALL READ (ECTA,12(NECTA),3,0,M)
IF (12(NECTA).EQ.ICQNM) GO TO 2
IF 4TH ARGUMENT OF READ EQUALS ONE THEN SKIP TO END OF RECORD
CCC WITHOUT TRANSFER OF DATA
CCC CALL READ(ECTA,12(NECTA),0,1,M)
130 CCC IF M NOT EQUAL TO ZERO THEN WE HAVE HIT THE END OF THE RECORD
CCC A FATAL ERROR
IF(M.NE.0)GOTO 500
CCC GO TO NEXT RECORD LOOKING FOR CQDNEM2
GO TO 1
135 CCC GET HERE WHEN CQDNEM2 RECORD IS FOUND
CCC SIL IS THE SCALAR INDEX LIST (SIMILAR TO AN ID ARRAY)
2 TRAIL(1)=SIL
CALL NDTRL(TRAIL)
140 CCC SIZE OF SIL = NDOP+1
NDOP=TRAIL(3)
NCP=TRAIL(2)
IF(NCP.NE.NDOP/6) GOTO 500
CCC OPEN 1 MORE GIMO BUFFER, TTTT MUST BE OPEN FOR
CCC WRITING WHILE ECTA IS OPEN FOR READING
RUF2=BUF1-1BUF
NCP=BUF2-1
CCC BECAUSE THERE ARE 7 WORDS PER CURRENT CQDNEM2 ELEMENT
CCC ONLY KEEP 7 WORDS OF CURRENT ECTA RECORD IN CORE AT A TIME
LECTA=7
CCC CHECK FOR ENOUGH SPACE IN GIMO TO PUT 2 BUFFERS
IF(MCORE.LT.NECTA+LECTA) GOTO 500
CCC THE NEXT SEVERAL LINES DETERMINE THE POSITION AND
CCC SIZE OF VARIOUS ARRAYS TO BE PLACED IN OPEN CORE
155 CCC ND = # OF OUTPUT POINTS FOR SUBROUTINE SPLINE (IN THIS
CCC CASE IT ALSO HAPPENS TO BE THE # OF ELEMENTS ON THE UPPER SURFACE)
NND=NECTA+LECTA
LND=2*ND
NTEX=NND+LND
160 CCC TEX IS MCP-BY-1 CONTAINS X COORDS OF GRID POINTS
CCC TEY IS MCP-BY-1 CONTAINS Y COORDS OF GRID POINTS
CCC ONE WORD FOR ALL GRID POINTS IS ALLOCATED, WHEN GOING THROUGH
CCC TOSOL (LIST OF TOP SURFACE ELEMENTS) UPPER SURFACE NODE COORD
CCC WILL BE FILLED IN, LEAVING THE REST FILLED WITH THE FLAG VALUE.
CCC THEN, LATER ON, ONLY THE UPPER SURFACE NODES WILL BE PUT INTO
CCC TEAR AND TEYR
LTEX=NCP
NTEY=NTEX+LTEX
LTFY=NCP
NITTT=NTEY+LTFY
170 LTTTT=NDOP

```

```

170  CCC      TTTT IS NDOF-NY-ND BUT ONLY ONE COLUMN IS BUILT AND PACKED
171  CCC      OUT AT A TIME. IN A COLUMN. PUT (1/4)AREA AT EACH UPPER SURFACE
172  CCC      ELEMENT'S TO ROW. THIS APPORTIONS 1/4 OF TOTAL ELEMENT
173  CCC      LOAD TO EACH CORNER NODE. NOTE! THIS ASSUMES APPROXIMATELY
174  CCC      SQUARE ELEMENTS.
175  CCC      NVCLS=NTTT+LTTTT
176  CCC      LVCLS=NDOF
177  CCC      ICORF=NVCLS+LVCLS
178  CCC      CHECK FOR ENOUGH SPACE IN GINO TO PUT THE ABOVE ARRAYS
179  CCC      IF(INCORE.LT.ICOREN) GO TO 500
180  CCC      FLAG=-10000.0
181  CCC      FROM HERE TO STATEMENT LABEL 7 ARRAYS NTEX,NTEY
182  CCC      AND,NVCLS ARE INITIALIZED
183  CCC
184  CCC      VCLS IS NDOF-NY-ONE VECTOR TO PUT INTO PG TO GET TOTAL LEFT.
185  CCC      IT IS ALSO USED TO PARTITION PCT.
186  CCC      IT CONTAINS A 1.0 IN EACH UPPER SURFACE NODE TO DOF
187  CCC
188  CCC      NTEX AND NTEY ARE INITIALIZED TO FLAG IN ORDER TO LATER ON
189  CCC      SPOT ANY VALUES IN THEM THAT HAVE NOT BEEN USED
190  CCC      USED ITEMS HAD BETTER BE UPPER SURFACE NODES FROM CONNECTIVITIES
191  CCC      OF TOSEL CODMEM2'S
192  CCC      DO 9 J=1,NDOF
193  CCC      Z(NTEX+J-1)=FLAG
194  CCC      Z(NTEY+J-1)=FLAG
195  CCC      9 CONTINUE
196  CCC      DO 7 J=1,LVCLS
197  CCC      Z(NVCLS+J-1)=0.0
198  CCC      7 CONTINUE
199  CCC
200  CCC      CALL COPEN(TTTT,ZCNUF2),1)
201  CCC      SPECIFIES CHARACTERISTICS OF ARRAYS FOR PACK SUBROUTINE
202  CCC      REAL SINGLE PRECISION INPUT FROM 2 ARRAY
203  CCC      TYPIM=1
204  CCC      REAL SINGLE PRECISION OUTPUT TO DATA BLOCK
205  CCC      TYPOT=1
206  CCC      INITIAL ROW POSITION
207  CCC      IRON=1
208  CCC      NROW WILL BE RESET PRIOR TO EACH CALL TO PACK SINCE MULTIPLE
209  CCC      MCR'S ARE MATRIX CONTROL BLOCKS FOR PACK ROUTINES
210  CCC      THEY WILL BE TRAILERS WHEN FINISHED, CONTENT IS THAT OF TRAILER
211  CCC      WORD 1 - GINO DR NAME
212  CCC      WORD 2 - # OF COLUMNS, ZERO INITIALLY, ACCUMULATED BY PACK
213  CCC      WORD 3 - # OF ROWS
214  CCC      WORD 4 - MATRIX SHAPE
215  CCC      WORD 5 - PRECISION      1 - REAL SINGLE PRECISION
216  CCC      WORD 6 - MAXIMUM DISTANCE BETWEEN NONZERO TERMS
217  CCC      WORD 7 - DENSITY
218  CCC      TRAILER FOR TTTT (TTTT SHOULD BE A NDOF*1 ARRAY)
219  CCC      MCR(1)=TTTT
220  CCC      MCR(2)=0
221  CCC      MCR(3)=LTTTT
222  CCC      MCR(4)=2
223  CCC      MCR(5)=1
224  CCC      MCR(6)=0
225  CCC      MCR(7)=0
226  CCC      LOOP 1 CALCULATES THE FOLLOWING DATA BLOCKS
227  CCC      TTTT,VCLS

```

```

230 CCC DO OVER ALL ELEMENTS OF TOSL. DO LENGTH SHOULD BE SIZE OF
    CCC TOSL (NOT # OF SPLINE OUTPUT POINTS)
    DO 3 1-1,ND
    CCC READS IN ONE CQDHEM2 ELEMENT AT A TIME FROM ECTA
    CALL READ(ECTA,Z(NECTA),7,0,M)
    CCC MAKE SURE THE CQDHEM2 ELEMENTS PRESENTED BY TOSL
    ARE IN ASCENDING CONSECUTIVE ORDER OR THE FOLLOWING MESSES UP
    CCC
    CCC IF THIS CQDHEM2 IS NOT ON THE UPPER SURFACE TRY THE NEXT ONE
    CCC CONTEXT OF THE 7 WORDS PER CQDHEM2 ELEMENT IS
    WORD 1 - EID, ELEMENT NUMBER
    WORD 2 - PID, PROPERTY ID
    WORDS 3,4,5,6 - CORNER NODE #'S
    WORD 7 - TH, MATERIAL ORIENTATION ANGLE.
    CCC FROM HERE TO LABEL 3 IS DONE ONCE FOR EACH CQDHEM2 ON THE UPPER
    CCC SURFACE
    IF(12(NECTA)-NE-12(NTOSL+1-1)) GO TO 4
    CCC CORNER NODE # FOR EACH CQDHEM2 ELEMENT
    IC1-12(NECTA+2)
    IC2-12(NECTA+3)
    IC3-12(NECTA+4)
    IC4-12(NECTA+5)
    CCC LATER ON (RIGHT AFTER LOOP 6) THE BCPA POINTERS (ICP#) ARE
    CCC CHECKED. IF THEY HAVEN'T CHANGED FROM ZERO THEN THE NODE
    CCC ISN'T IN BCPA, A BIG ERROR
    ICP1=0
    ICP2=0
    ICP3=0
    ICP4=0
    CCC LOCATES APPROPRIATE NODES IN GPLA
    CCC SEARCH GPLA - LIST OF ALL GRID POINT IDENTIFIERS FOR EACH
    CCC CORNER NODE. THE POSITION IN GPLA IS THE POINTER TO THE
    CCC ID, X, Y, Z OF THE NODE IN BCPA
    DO 6 J=1,LCPLA
    K=GPLA+J-1
    IF(12(NECTA)-12(K)) IC1=NRGPA+(J-1)*4
    IF(12(NECTA)-12(K)) IC2=NRGPA+(J-1)*4
    IF(12(NECTA)-12(K)) IC3=NRGPA+(J-1)*4
    IF(12(NECTA)-12(K)) IC4=NRGPA+(J-1)*4
    CCC
    CCC 6 CONTINUE
    IF NODE IN ECTA IS NOT IN GPLA, THE WORLD IS UPSIDE DOWN!
    IF(12(NECTA)-12(K)) GO TO 500
    IF(12(NECTA)-12(K)) GO TO 500
    IF(12(NECTA)-12(K)) GO TO 500
    IF(12(NECTA)-12(K)) GO TO 500
    PUTS 1.0 INTO EACH UPPER SURFACE NODE Z DOF INTO VGLS
    Z(NVGLS+6*IC1-4)=1.0
    Z(NVGLS+6*IC2-4)=1.0
    Z(NVGLS+6*IC3-4)=1.0
    Z(NVGLS+6*IC4-4)=1.0
    CCC GET X,Y COORDS OF CORNER NODES FROM BCPA
    X1=Z(12(NECTA)+1)
    Y1=Z(12(NECTA)+2)
    X2=Z(12(NECTA)+3)
    Y2=Z(12(NECTA)+4)
    X3=Z(12(NECTA)+5)
    Y3=Z(12(NECTA)+6)

```

11/03/82 20.07.66

FTN 4.8+564

74/74 OPT-1

SUBROUTINE GIAS

```

      X4=Z(IGP4+1)
      Y4=Z(IGP4+2)
      CCC  CENTROIDAL COORDINATES OF QUAD
      CCC  THE SPLINE OUTPUT POINTS
      Z(NNH+2*1-2)=(X1+X2+X3+X4)/4.
      Z(NHD+2*1-1)=(Y1+Y2+Y3+Y4)/4.
      CCC  AREA OF ELEMENT
      AREA=.5*(X3-X1)*(Y4-Y2)-(Y3-Y1)*(X4-X2))
      CCC  ZEROS OUT THE COLUMNS
      DO 11 J=1,LTITTT
      11 Z(NITTT+J-1)=0.0
      CCC  .25 LOAD ON MEMBRANES TO THEIR NODES IN Z DOF FOR EACH NODE.
      CCC  IN TITTT
      CCC  PRESUMES ALMOST SQUARE ELEMENTS
      Z(NITTT+6*IC1-4)=AREA*.25
      Z(NITTT+6*IC2-4)=AREA*.25
      Z(NITTT+6*IC3-4)=AREA*.25
      Z(NITTT+6*IC4-4)=AREA*.25
      CCC  LENGTH OF A COLUMN IN TITTT FOR PACK
      NROH=LTITTT
      CCC  PAGES TITTT ONE COLUMN AT A TIME
      CALL PACK(Z(NITTT),TITTT,MCBB)
      CCC  USED FOR VTH,VTR (THE MOMENT ARMS)
      CCC  THE MOMENT ARMS OF EACH NODE ARE THE COORDINATES OF THE NODE
      Z(MTEX-1+IG1)=X1
      Z(MTEX-1+IG2)=X2
      Z(MTEX-1+IG3)=X3
      Z(MTEX-1+IG4)=X4
      Z(MTEY-1+IG1)=Y1
      Z(MTEY-1+IG2)=Y2
      Z(MTEY-1+IG3)=Y3
      Z(MTEY-1+IG4)=Y4
      3 CONTINUE
      CALL CLOSE(ECTA,1)
      CALL CLOSE(TITTT,1)
      CALL WRITRL(MCRP)
      CCC  SET UP A MATRIX CONTROL BLOCK FOR PACKING VGLS
      CCC  TRAILER VGLS (SIZE = NDOF*1)
      MCRA(1)=VGLS
      MCRA(2)=0
      MCRA(3)=LVGLS
      MCRA(4)=2
      MCRA(5)=1
      MCRA(6)=0
      MCRA(7)=0
      CCC  SET COLUMN LENGTH FOR PACKING VGLS
      FROM=LVGLS
      CALL GOPEN(VGLS,Z(RUP1),1)
      CCC  PAGES THE ONE COLUMN OF VGLS
      CALL PACK(Z(NVGLS),VGLS,MCRA)
      CALL CLOSE(VGLS,1)
      CCC  CALL WRITRL(MCRA)
      CCC  NOW REDUCE TTX TO JUST UPPER SURFACE NODES AND PACK OUT
      CCC  AS VTH
      CCC  REUSE THE SPACE IN OPEN CORE USED BY TITTT, VGLS
      CALL GOPEN(VTH,Z(RUP1),1)
      CALL GOPEN(VTH,Z(RUP2),1)

```

11/03/82 20.07.46

PTN 4.8-544

74/74 OPT-1

SUBROUTINE GIAS

```

345 CCC OFFINES LENGTH AND POSITION OF TEXR AND TETR (MODIFIED
    CCC TRY AND TRY), TO BE PACKED AS VTM AND VTR
    CCC
    CCC ACTUAL # OF NODES ON UPPER SURFACE IS UNKNOWN, IT MUST
    CCC <= NCP. THE VARIABLE NC WILL COUNT THEM IN THE FOLLOWING LOOP.
    CCC FOR NOW JUST SET ASIDE NCP WORDS, THEN AFTER REDUCTION LOOP,
    CCC PACK OUT FIRST NC WORDS.
    NTEYR=NTTTT
    NTEYR=NTTEYR+NCP
    NC=0
    CCC LOOP 13 CALCULATES THE VTM AND VTR MATRICIES
    CCC FILTERS OUT THE UPPER SURFACE VALUES OF TEX AND TRY
    DO 13 J=1,NCP
    IF(Z(NTTEYR+J-1)-EQ-FLAG) GO TO 13
    Z(NTTEYR+NC)=Z(NTTEYR+J-1)
    Z(NTTEYR+NC)=Z(NTTEYR+J-1)
    NC=NC+1
    13 CONTINUE
    CCC IF NC=0 THEN FATAL ERROR, NO UPPER SURFACE NODES
    IF(NC.EQ.0) GOTO 500
    CCC TRAILER OF VTM (SIZE = NC)
    NROW=NC
    MCRA(1)=VTM
    MCRA(2)=0
    MCRA(3)=NC
    MCRA(4)=2
    MCRA(5)=1
    MCRA(6)=0
    MCRA(7)=0
    CCC TRAILER OF VTR (SIZE = NC)
    MCRB(1)=VTR
    MCRB(2)=0
    MCRB(3)=NC
    MCRB(4)=2
    MCRB(5)=1
    MCRB(6)=0
    MCRB(7)=0
    CCC PACKS NTEYR AND NTEYR AS VTM AND VTR
    CALL PACK(Z(NTTEYR),VTM,MCRA)
    CALL PACK(Z(NTTEYR),VTR,MCRB)
    CALL CLOSE(VTM,1)
    CALL CLOSE(VTR,1)
    CALL WRTTTL(MCRA)
    CALL WRTTTL(MCRB)
    CCC IDP'S OF AERO BOXES' CORNERS START AT MIL+1
    MIL=1000000
    1=1
    CCC MCB EQUALS THE NUMBER OF CHORDWISE BOXES
    CCC SETS POSITION AND SIZE IN OPEN CORE FOR AERO INPUT POINTS
    CCC FOR SUBROUTINE SPLINE
    LSTND=0
    NESTBY=1
    CCC LOOP UNTIL STATEMENT LABEL 14 IS SIMILAR TO LOOP 3 BUT FOR
    CCC AERO POINTS
    CCC THE PLAN: BUILD COORDS OF SPLINE INPUT POINTS WHICH ARE THE
    CCC 1/2 STAY, 1/4 CHORD OF THE AERO BOXES (UNDER THE DELTA CP IS
    CCC PRELIMED TO LIVE). GET COORD OF THIS POINT FROM THE COORDS OF

```



```

460      NND=1
      LNI=2*NI
      DO 210 I=1,LNI
      210 Z(NND+LND+I-1)=Z(NNI+I-1)
      NNI=NND+LND
      CCC
      CCC
      CCC
465      TRAILER GTAK (SIZE = # OF AERO BOXES TIMES NUMBER OF
      STRUCTURAL BOXES)
      PRESUMES CONSTANT PRESSURE ASSUMPTION ON STRUCTURAL BOXES
      MCRA(1)=GTAK
      MCRA(2)=0
      MCRA(3)=ND
      MCRA(4)=2
      MCRA(5)=1
      MCRA(6)=0
      MCRA(7)=0
      NCTAK=NNI+NI*2
      FLAG FOR TRANSPOSITION: KT=0 DO NOT TRANSPOSE OUTPUT
      KT=0
      CCC
475      SYMMETRY FLAG FOR X-DIRECTION: KX=0 GENERAL
      KX=0
      CCC
      SYMMETRY FLAG FOR Y-DIRECTION: KY=0 GENERAL
      KY=0
      CCC
480      FLAG FOR SLOPES: KD=0 INTERPOLATION ONLY
      KD=0
      CCC
      ATTACHMENT FLEXIBILITY: REAL >= 0.0
      DE=0.0
      CCC
      RETURNS RUF2
      NCORE=RUF1-1
      CCC
485      SPECIFIES THE AMOUNT OF OPEN CORE REQUIRED TO RUN SSPLIM
      NSCF=(NI+3)*2+3*(NI+3)+NI*ND+NI*(NI+3)+2
      THIS CHECKS OPEN CORE TO SEE IF ENOUGH IS AVAILABLE
      IF(NCORE.LT.(NSCF+NCCTAK)) GO TO 500
      CALL SSPLIM(NI,Z(NNI),ND,Z(NND),KX,KY,KD,DZ,Z(NCTAK),NSCF,
      *ISING)
      RESETS AMOUNT OF CORE LEFT ASIDE FOR BUFFERS (# OF BUFFERS
      REQUIRED CHARGES FROM 1 TO 2)
      CCC
495      NCORE=RUF2-1
      IF(NCORE.LT.(NCTAK+NI*ND)) GO TO 500
      CALL COPEN(GTAK,Z(RUF1),1)
      NROW=ND
      CCC
500      PACKS NI COLUMNS OF GTAK, EACH COLUMN IS ND LONG
      DO 16 J=1,NI
      CALL PACK(Z(NCTAK+(J-1)*ND),GTAK,MCRA)
      16 CONTINUE
      CALL CLOSE(GTAK,1)
      CALL UNTRL(MCRA)
      CALL COPEN(VTMA,Z(RUF1),1)
      CALL COPEN(VTRA,Z(RUF2),1)
      BEGIN A NEW TEXT AND TEXT (THIS TIME FOR AERO POINTS) AT
      CCC
      CCC
      POSITIONS NCTAK AND NCTAK+NI
      NTEX=NCTAK
      LTEX=NI
      LTEX=NI
      LTEX=NI
      IF(NCORE.LT.(NTEX+LTEX)) GO TO 500
      CCC
      LOOP 17 CALCULATES THE VTMA AND VTRA MATRICES

```

```

515      DO 17 I=1,NI
          CCC      COORDINATES OF 1/4 POINTS OF AFRO BOXES
          7(NTEX+1-1)=Z(NKI+2*1-1-1)
          Z(NTEX+1-1)=Z(NNI+2*1-1)
          17 CONTINUE
          NROW=NI
520      CCC      TRAILER VTMA (SIZE = NI*1)
          MCRA(1)=VTMA
          MCRA(2)=0
          MCRA(3)=NI
          MCRA(4)=2
          MCRA(5)=1
          MCRA(6)=0
          MCRA(7)=0
530      CCC      TRAILER VTVA (SIZE = NI*1)
          MCB(1)=VTVA
          MCB(2)=0
          MCB(3)=NI
          MCB(4)=2
          MCB(5)=1
          MCB(6)=0
          MCB(7)=0
535      CCC      PACKS BOTH VTMA AND VTVA AS NI BY 1 ARRAYS
          CCC      NOTICE BOTH PACKS USE SAME NROW VALUE
          CALL PACK(Z(NTEX),VTMA,MCRA)
          CALL PACK(Z(NTEY),VTVA,MCBB)
          CALL CLOSE(VTMA,1)
          CALL CLOSE(VTVA,1)
          CALL WRTTTL(MCRA)
          CALL WRTTTL(MCBB)
          MAIDMT=1
545      DO 18 I=1,NI
          Z(NAIDMT+1-1)=1.0
          18 CONTINUE
          CALL GOPEN(AIDMT,Z(BUF1),1)
          TRAILER AIDMT (VECTOR OF ONES NI LONG)
          AIDMT IS USED IN CALCULATING THE CL OF THE AFRO MODEL
          MCRA(1)=AIDMT
          MCRA(2)=0
          MCRA(3)=NI
          MCRA(4)=2
          MCRA(5)=1
          MCRA(6)=0
          MCRA(7)=0
          NROW=NI
          CALL PACK(Z(NAIDMT),AIDMT,MCRA)
          CALL CLOSE (AIDMT,1)
          CALL WRTTTL(MCRA)
          CALL GOPEN(SAS,Z(BUF1),1)
          COMPLEX
          TYPLOT=1
          CCC
          CCC      BEGIN AT BEGINNING OF OPEN CORE
          NSAS=1
          NROW=2*NI
          CCC      RELIEVES ONE (TYPE COMPLEX) CAN BE BUILT IN CORE AS TWO
          CCC      CONSECUTIVE VALUES
          NTL=NI*ERROR

```

CCC ENTIRE SAS IS BUILT IN CORE THEN PACKED, COULD BE BUILT
 CCC ONE COLUMN AT A TIME BUT BY NOW THERE IS LOTS OF ROOM
 CCC ZFPOS OUT SAS

575 DO 19 I=1,NL
 Z(NSAS+I-1)=0.0
 10 CONTINUE
 L=1

580 DO 20 I=1,NI
 REAL PART
 Z(NSAS+L+I-1+I-2)=1.0
 CCC COMPLEX PART
 Z(NSAS+L+I-1+I-1)=1.0
 L=L+NROW

585 20 CONTINUE
 CCC TRAILER SAS
 MCRA(1)=SAS
 MCRA(2)=0
 MCRA(3)=NROW
 MCRA(4)=2
 MCRA(5)=3
 MCRA(6)=0
 MCRA(7)=0
 L=1

595 DO 21 I=1,NI
 CALL PACK(Z(NSAS+L-1),SAS,MCRA)
 L=L+NROW
 21 CONTINUE

600 CALL CLOSE(SAS,1)
 CALL UNTRL(MCRA)
 RETURN
 CCC GENERAL ERROR EXIT, NO CAUSES GIVEN
 500 N=-8

605 NSUB=16
 CALL MESSAGE(N,NSUB,NAMT)
 RETURN
 END

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 1 GIAS

VARIABLES	SN	TYPE	RELOCATION		
1531 AJUNT		REAL	1513 ACPT		INTEGER
1517 RCPA		INTEGER	1632 AREA		REAL
1550 RUF2		INTEGER	1547 RUF1		INTEGER
1666 DZ		REAL	1551 RUF3		INTEGER
1606 FIAC		REAL	1515 ECTA		INTEGER
1522 CIAY		INTEGER	1516 GPLA		INTEGER
1653 IAP1		INTEGER	1610 I		INTEGER
1655 IAP3		INTEGER	1656 IAP2		INTEGER
1647 IAI		INTEGER	1656 IAP4		INTEGER
			1650 IAI2		INTEGER
					*UNDEF

125 2
0 3
0 6
0 11
0 15
0 16
0 19
0 28
0 210
1165 500

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES	EXT REFS	EXITS	NOT INNER
167	9	J	194 197	38	INSTACK			
177	7	J	198 200	28	INSTACK			
216	3	I	231 318	2028				
243	6	J	262 268	268	OPT			
353	11	J	295 296	28	INSTACK			
473	13	J	355 360	68	INSTACK			
5-2	14	I	403 447	1278				
553	28	J	407 444	1078				
576	15	KK	417 423	268	OPT			
675	200	I	455 456	38	INSTACK			
711	210	I	459 460	28	INSTACK			
760	16	J	499 501	118				
1017	17	I	514 518	38	INSTACK			
1062	18	I	545 547	28	INSTACK			
1122	19	I	574 576	28	INSTACK			
1130	20	I	578 584	78	INSTACK			
1150	21	I	594 597	108				

COMMON BLOCKS	LENGTH
/	1
SYSTEM	1
PACKX	5
ZOPEN	1

STATISTICS	PROGRAM LENGTH	17200	981
CM LAFELD COMMON LENGTH	78		7
CM BLANK COMMON LENGTH	18		1
520008 CM USED			

```

1      CCC
   CCC
   CCC
   CCC
   CCC
      SUBROUTINE INIT
   CCC
   CCC
   CCC
      READS DATA BLOCKS FROM "PUNCH" TAPE INTO MEMORY
   COMMON/SYSTEM/IBUF
   COMMON/2DATA/ZACPT(49),ZRCFA(164),ZTOSL(4),ZCPLA(41),ZSIL(18)
   1,ZECTA(64,4)
   COMMON/OTRAIL/TACPT(6),TRCFA(6),TTOSL(6),TCPLA(6),TSIL(6),
   ITECTA(6)
   15  INTEGER TACPT,TTOSL,TECTA,TCPLA,TRCFA,TSIL,ZTOSL,
   ZCPLA,ZSIL,ZECTA,IZACPT(49)
   REAL ZACPT,ZRCFA
   EQUIVALENCE (IZACPT(49),ZACPT(49))
   INUF=8
      READ ACPT AND TRAILER
   CCC
   CCC
   CCC
   930  READ(5,930)(TACPT(I),I=1,6)
      FORMAT(24X,6I8)
   25  READ(5,500)
      FORMAT(18X)
   500  READ(5,931)(IZACPT(I),I=1,2)
      FORMAT(53X,13,13X,13)
   931  READ(5,932)IZACPT(3),IZACPT(4),ZACPT(5),IZACPT(6)
      FORMAT(23X,11,15X,11,16.9,15X,11)
   932  IF(IZACPT(2)-EQ.3)GOTO 501
      IF(IZACPT(2)-EQ.3)GOTO 502
   970  READ(5,970)IZACPT(7),(ZACPT(I),I=8,10)
      FORMAT(23X,11,3E16.9)
      GOTO 510
   502  READ(5,503)IZACPT(1),I=7,10)
   503  FORMAT(8X,4(15X,11))
      GOTO 510
   501  READ(5,505)(IZACPT(1),I=7,9),ZACPT(10)
   505  FORMAT(23X,11,2(15X,11),E16.9)
   510  CONTINUE
      IF(IZACPT(2)-EQ.3)GOTO 511
   971  READ(5,971)(ZACPT(I),I=11,14)
      FORMAT(8X,4E16.9)
      GOTO 512
   511  READ(5,970)IZACPT(11),(ZACPT(I),I=12,14)
   512  CONTINUE
      IF(IZACPT(2)-EQ.3)GOTO 520
      IF(IZACPT(2)-EQ.3)GOTO 513
   50  READ(5,972)ZACPT(15),ZACPT(16),IZACPT(17),IZACPT(18)
      FORMAT(8X,2E16.9,2(15X,11))
   972  READ(5,970)IZACPT(19),(ZACPT(I),I=20,22)
      GOTO 530
   513  DO 514 J=1,2
   514  READ(5,971)(ZACPT(14+I+(J-1)*4),I=1,4)
      GOTO 530
   520  READ(5,971)(ZACPT(1),I=15,18)

```

```

60      530      READ(5,505)((IZACPT(I),I=19,21),ZACPT(22))
          CONTINUE
          NP=4
          IF(IZACPT(2)-EQ.2)NP=5
          IF(IZACPT(2)-EQ.3)GOTO 515
          DO 10 J=1,NP
            READ(5,971)((ZACPT(22+I+(J-1)*4),I=1,4)
            IF(IZACPT(2)-EQ.2)GOTO 526
            READ(5,973)((ZACPT(1),I=39,40)
            FORMAT(8X,2E16.9)
            GOTO 526
          515      READ(5,516)ZACPT(23),(IZACPT(1),I=24,26)
          516      FORMAT(8X,E16.9,3(15X,11))
            READ(5,970)IZACPT(27),(ZACPT(1),I=28,30)
            DO 517 J=1,4
          517      READ(5,971)((ZACPT(30+I+(J-1)*4),I=1,4)
            READ(5,518)((ZACPT(1),I=47,49)
            FORMAT(8X,2E16.9)
            READ(5,500)
          70      515      READ TOSEL AND TRAILER
          516      FORMAT(19H BEGINNING OF TOSEL)
            READ(5,930)((TOSEL(I),I=1,6)
            READ(5,500)
            DO 43 J=1,4
          80      940      READ(5,940)((TOSEL(I),I=1,4)
            940      FORMAT(24X,418)
            940      READ PCTA AND TRAILER
            940      READ(5,930)((PCTA(I),I=1,6)
            940      READ(5,500)
            DO 43 J=1,4
          90      43      ZECTA(I,J)=1
            DO 15 J=1,4
            950      READ(5,950)((ZECTA(I,J),I=1,6)
            950      FORMAT(24X,618)
            IF (J.NE.1) GOTO 50
            DO 951 M=1,6
          100      951      READ(5,952)((ZECTA(M-1)+I*6,1),I=1,8)
            952      FORMAT(8X,818)
            READ(5,953)((ZECTA(1,1),I=55,59)
            953      FORMAT(8X,518)
            GOTO 15
          105      50      IF (J.NE.2) GOTO 51
            DO 954 M=1,4
          110      954      READ(5,952)((ZECTA(M-1)+I*6,2),I=1,8)
            954      FORMAT(8X,18)
            GOTO 15
            51      IF (J.NE.3) GOTO 52
            DO 956 M=1,7
          52      956      READ(5,952)((ZECTA(M-1)+I*6,3),I=1,8)
            956      FORMAT(8X,3)
            GOTO 15
            52      CONTINUE

```

```

135      DO 957 M=1,4
136          READ(5,952)(ZEGTA(R*(M-1)+1+6,4),I=1,R)
137          READ(5,953)ZEGTA(39,4)
138      CONTINUE
139      READ GPLA AND TRAILER
140      READ(5,930)(TGPLA(I),I=1,6)
141      READ(5,500)
142      READ(5,961)(ZGPLA(I),I=1,6)
143      FORMAT(24X,610)
144      NR=3
145      IF(IZACPT(2).NE.1)NR=4
146      DO 16 J=1,NR
147          READ(5,952)(ZGPLA(R*(J-1)+6+1),I=1,R)
148          IF(IZACPT(2).EQ.2)GOTO 540
149          IF(IZACPT(2).EQ.3)GOTO 541
150          READ(5,962)(ZGPLA(I),I=31,36)
151          GOTO 550
152      READ(5,542)(ZGPLA(I),I=39,41)
153      FORMAT(8X,310)
154      GOTO 550
155      READ(5,543)ZGPLA(39)
156      FORMAT(8X,18)
157      CONTINUE
158      FORMAT(8X,610)
159      READ NGPA AND TRAILER
160      READ(5,930)(TBGPA(I),I=1,6)
161      READ(5,500)
162      READ(5,931)ZBGPA(1),ZBGPA(2)
163      READ(5,963)(ZBGPA(I),I=3,6)
164      FORMAT(21X,F3.1,E16.9,13X,F3.1,13X,F3.1)
165      DO 17 J=1,5
166          READ(5,964)(ZBGPA(J-1)+1+6),I=1,4)
167          FORMAT(21X,F3.1,E16.9,13X,F3.1,E16.9)
168          DO 18 J=1,11
169              READ(5,965)(ZBGPA(J-1)+4+1+26),I=1,4)
170              FORMAT(8X,2E16.9,13X,F3.1,E16.9)
171          READ(5,966)(ZBGPA(I),I=71,74)
172          FORMAT(8X,2E16.9,13X,F3.1,13X,F3.1)
173          READ(5,964)(ZBGPA(I),I=75,78)
174          READ(5,964)(ZBGPA(I),I=79,82)
175      DO 19 J=1,3
176          READ(5,965)(ZBGPA(J-1)+4+1+82),I=1,4)
177          READ(5,964)(ZBGPA(I),I=95,98)
178      NR=11
179      IF(IZACPT(2).EQ.2)NR=14
180      IF(IZACPT(2).EQ.3)NR=16
181      DO 20 J=1,NR
182          READ(5,965)(ZBGPA(J-1)+4+1+98),I=1,4)
183          IF(IZACPT(2).EQ.2)GOTO 560
184          IF(IZACPT(2).EQ.3)GOTO 561
185          READ(5,967)ZBGPA(163),ZBGPA(164)
186          GOTO 570
187      READ(5,967)ZBGPA(163),ZBGPA(164)

```

```

COTO 570
560 READ(5,967)ZRCPA(155),ZRCPA(156)
570 CONTINUE
967 FORMAT(8X,2E16.9)
    READ(5,500)
    READ(5,500)
    IF(IZACPT(2).EQ.1) READ(5,500)
    READ STL AND TRAILFR
    READ(5,930)(TSIL(I),I=1,6)
    READ(5,500)
    READ(5,961)(ZSIL(I),I=1,6)
    READ(5,969)(ZSIL(1+6),I=1,8)
    READ(5,968)(ZSIL(1+14),I=1,4)
    FORMAT(8X,418)
968
969
    RETURN
    END
    
```

SYMBOLIC REFERENCE MAP (R-1)

ENTRY POINTS

VARIABLES	SN	TYPE	RELOCATION	0	IRUF	INTEGER	SYSTEM
1516 I	0	IZACPT	ARRAY ZDATA	1517 J	INTEGER	INTEGER	
1521 N	0	TACPT	ARRAY DTRAIL	1520 NR	INTEGER	INTEGER	DTRAIL
36 TCCTA	0	TSIL	ARRAY DTRAIL	22 TCPLA	INTEGER	INTEGER	DTRAIL
30 TSIL	0	ZACPT	ARRAY ZDATA	14 TIOSEL	REAL	REAL	ZDATA
424 ZECTA	0	ZSIL	ARRAY ZDATA	61 ZRCPA	INTEGER	INTEGER	ZDATA
402 ZSIL	0	ZSIL	ARRAY ZDATA	331 ZCPLA	INTEGER	INTEGER	ZDATA
				325 ZTIOSEL	INTEGER	INTEGER	ZDATA

FILE NAMES

TAPE 5

STATEMENT LABELS

1057 3	FMT	NO REFS
0 16		0 10
0 19		0 17
241 50		0 20
637 500	FMT	267 51
703 503	FMT	24 501
33 511		714 505
0 514		35 512
0 517		127 515
156 526		1051 518
404 541		73 530
411 550		1252 542
		553 540

340 15	
0 18	
0 43	
315 52	
21 502	
26 510	
45 513	
1024 516	
67 520	FMT
407 540	
1261 545	FMT
550 561	

FTN 4.-R+564 11/03/82 20.07.46

74/74 OPT-1

SUBROUTINE INIT
STATEMENT LABELS

555 570		631 930	FMT	646 931	FMT
661 932	PMT	1101 940	PMT	1121 950	PMT
0 951		1132 952	PMT	1141 953	PMT
0 954		1157 955	FMT	0 956	
0 957		1227 961	PMT	1263 962	PMT
1311 963	FMT	1325 964	FMT	1340 965	PMT
1351 966	FMT	1434 967	FMT	1512 968	PMT
1514 969	FMT	673 970	PMT	725 971	PMT
745 972	FMT	1014 973	PMT		

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
46	514	J	54 55	218	EXT REFS NOT INNER
51		I	55 55	128	EXT REFS
102	10	J	63 64	218	EXT REFS NOT INNER
105		I	64 64	128	EXT REFS
134	517	J	72 73	218	EXT REFS NOT INNER
137		I	73 73	128	EXT REFS
173	43	J	90 92	128	NOT INNER
200	43	I	91 92	28	
206	15	J	93 118	1358	EXT REFS NOT INNER
217	951	M	97 98	208	EXT REFS NOT INNER
222		I	98 98	118	EXT REFS
244	954	M	104 105	218	EXT REFS NOT INNER
247		I	105 105	128	EXT REFS
272	956	M	110 111	218	EXT REFS NOT INNER
275		I	111 111	128	EXT REFS
316	457	M	115 116	218	EXT REFS NOT INNER
321		I	116 116	128	EXT REFS
355	16	J	128 129	218	EXT REFS NOT INNER
360		I	129 129	128	EXT REFS
422	17	J	149 150	208	EXT REFS NOT INNER
425		I	150 150	118	EXT REFS
443	18	J	152 153	208	EXT REFS NOT INNER
446		I	153 153	118	EXT REFS
471	19	J	159 160	208	EXT REFS NOT INNER
474		I	160 160	118	EXT REFS
522	20	J	165 166	208	EXT REFS NOT INNER
525		I	166 166	118	EXT REFS
600		I	186 186	78	EXT REFS
613		I	187 187	78	EXT REFS

INSTACK

COMMON FLOCKS LENGTH
SYSTEM 1
ZDATA 532
DTRAIL 36

STATISTICS
PROGRAM LENGTH 15228 850
CM LABELED COMMON LENGTH 10718 560
520004 CM USED


```

1      CCC
2      CCC
3      CCC
4      CCC
5
10     SUPPORTIF SSPLIN(M1,XV1,KD,XVD,KX,KY,KD,VT,DZ,C,NCORE,1SMG)
      LOGICAL LX,LY,LONE,IKD,IKT
      DIMENSION C(1),XV1(1),XVD(1),NAME(2)
      REAL DET
      DATA NAME/4HSSPL,4HIN /
      LONE = -.TRUE.
      LX = -.TRUE.
      LY = -.TRUE.
      IKT = -.FALSE.
      IKD = -.FALSE.
      IF(KY.LT.0.OR.KX.LT.0) LONE = .FALSE.
      IF(VY.LT.0.OR.KX.GT.0) LX = .FALSE.
      IF(KY.GT.0.OR.KX.LT.0) LY = .FALSE.
      N = NI
      IF(LONE) N=N+1
      IF(LX ) N=N+1
      IF(LY ) N=N+1
      EX = FLOAT(KX)
      EY = FLOAT(KY)
      IF(KT.EQ.1) IKT = .TRUE.
      IF(KD.EQ.1) IKD = .TRUE.
      NS = ND*(1+IKD)
      CORE NEEDED
20     C
30     C
35     C
      NEEDED = A C NRMN1 R + 3*N C
      IF(IKT) NEEDED = NEEDED + NRMN + NI*N C
      IF(.NOT.IKT) NEEDED = NEEDED + NI*N + MAX0(N*N,NRMN) A OR B
      IF(.NOT.D.GT.NCORE) CALL MESAGE(-8,0,NAME)
      IS = NCORE - 3*N -1
      IC = 1
40     C
      IF KT = 1 COMPUTE R THEN A THEN C IN A SPACE
50     C
      IF VT = 0 COMPUTE C THEN A THEN R IN A SPACE
      NT = 2*NI
      IF(.NOT.IKT) GO TO 65
      GO TO 95
      COMPUTE TO A MATRIX
      I R = 1A
      ZFRD A
      I J = K+1
      IK = 11 + N*N
      DO 5 I = 1,11

```

11/07/82 20.07.46

FTN 4.PASS64

SYNOPSIS 55PLIN 74/74 OPT=1

```

5  C(I) = 0.0
6  I1 = 1
7  IF = 0
8  DO 60 I = 1, NT, 2
9  K = P+K
10 JJ = I/2
11 DO 20 J = 1, NT, 2
12 K = P+1
13 JJ = JJ + 1
14 SUM = 0.0
15 V4 = (XV1(I) - XV1(J)) **2
16 XP = (XV1(I) + XV1(J)) **2
17 Y4 = (XV1(I+1) - XV1(J+1)) **2
18 YP = (XV1(I+1) + XV1(J+1)) **2
19 T1 = X4+Y4
20 T2 = X4+Y4
21 T3 = X4+Y4
22 T4 = X4+Y4
23 IF(T1.NE.0.0) SUM = T1 + ALOG(T1)
24 IF(T2.NE.0.0.AND.KX.NE.0) SUM = SUM + (T2*ALOG(T2)*EX)
25 IF(T3.NE.0.0.AND.KY.NE.0) SUM = SUM + (T3 + ALOG(T3) * FY)
26 IF(T4.NE.0.0.AND.KY.NE.0.AND.KX.NE.0) SUM = SUM + (T4*ALOG(T4)*EX*FY)
27 IF(J.EQ.1) GO TO 10
28 C(V) = SUM
29
30 SYMPTRY TERM
31
32 KX = Y + (N-1)*(JJ-1)
33 C(KX) = SUM
34 GO TO 20
35 10 C(K) = SUM + DZ
36 KX = K
37 20 CONTINUE
38 INP = 0
39 IF(.NOT.LONE) GO TO 30
40 INP = INP + 1
41 C(K+INP) = 1.0
42 C(K+INP*N) = 1.0
43 30 IF(.NOT.LX) GO TO 40
44 INP = INP + 1
45 C(K+INP) = XV1(I)
46 C(K+INP*N) = XV1(I)
47 40 IF(.NOT.LY) GO TO 50
48 INP = INP + 1
49 C(K+INP) = XV1(I+1)
50 C(K+INP*N) = XV1(I+1)
51 50 IF = 11 + INP
52 I1 = I1 + 1
53 60 CONTINUE
54
55 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110

```

SUBROUTINE SSPLIN 74/74 OPT=1

```

115      GO TO 70
      C
      C
      C MATRIX COLUMN STORED
      C
      45 IC = NR*NI
      70 DO 90 I = 1,NI
      DO 80 J = 1,N
      IC = IC+1
      G(IC) = 0.0
      IF(I.EQ.J) G(IC) = 1.0
      80 CONTINUE
      90 CONTINUE
      IF(IFT) GO TO 170
      NC = NI
      IA = IC
      GO TO 1
      C
      C
      C P MATRIX COLUMN STORED
      C
      95 IB = NR*NI
      MC = IB+1
      GO TO 110
      100 IR = IA
      110 NF = 2*ND
      K = IR+1
      DO 160 J = 1,NR,2
      DO 120 I = 1,NI,2
      IR = IR+1
      ALTI = 0.0
      ALTI2 = 0.0
      ALTI3 = 0.0
      ALTI4 = 0.0
      XM = XYD(J)-XYI(1)
      XP = XYI(1)+XYD(J)
      YM = XYD(J+1)-XYI(1+1)
      YP = XYI(1+1)+XYD(J+1)
      T1 = XM*YM + YM*YM
      T2 = XP*XP + YM*YM
      T3 = XM*XM + YP*YP
      T4 = XP*XP + YP*YP
      IF(T1.NE.0.0) ALTI = ALOC(T1)
      IF(T2.NE.0.0.AND.PX.NE.0)ALTI2 = ALOC(T2)
      IF(T3.NE.0.0.APD.KY.NE.0)ALTI3 = ALOC(T3)
      IF(T4.NE.0.0.AND.KX.NE.0)ALTI4 = ALOC(T4)
      C(IR) = T1*ALTI + T2*ALTI2*EX + T3*ALTI3*EY + T4*ALTI4*EX*EY
      IF(.NOT.IPD) GO TO 120
      IV = IR + N
      C(IV) = 2.0*(YM*(1.0+ALTI) + XP*(1.0+ALTI2)*EY +
      * XM*(1.0+ALTI3)*EY + XP*(1.0+ALTI4)*EX*EY)
      120 CONTINUE
      75P = 0
      130 IF(.NOT.IOR) GO TO 135
      131 P = PX+1
      C(IR+1IP) = 1.0
      IF(IV) C(IR+1PP+E) = 0.0
      135 IF(.NOT.IX) GO TO 140

```

AD-A124 662

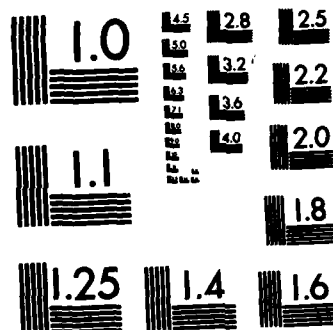
STATIC AEROELASTIC ANALYSIS OF FLEXIBLE WINGS VIA
NASTRAN PART I (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING K JONES
DEC 82 AFIT/GAE/AA/82D-16-PT-1 F/G 1/3

2/2

UNCLASSIFIED

NL

							END						
							FILED						
							DTIC						



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

11/03/82 20.07.46

FTN 4.8+564

74/74 OPT-1
RELOCATION
P.P.

SUBROUTINE SSPLIN

VARIABLES SN TYPE
0 NI INTEGER
600 NI INTEGER
615 T1 REAL
617 T3 REAL
611 IM REAL
0 XVD REAL
613 YM REAL

627 NR
610 SUM
616 T2
620 T4
612 TP
0 XT1
614 TP

P.P.

ARRAY

REAL

CHMATS
MESSAGE

EXTERNALS TYPE
ALOC REAL
INVERS 8
LIBRARY 1

ARCS
1
LIBRARY 8

INLINE FUNCTIONS TYPE
FLOAT REAL
1 INTRIN

ARCS
1
INTRIN

MAXO INTEGER 0 INTRIN

STATEMENT LABELS

104 1
216 20
245 50
275 70
320 95
425 120
457 150
503 180
0 5
227 30
0 60
0 80
324 100
437 130
0 160
514 1000
212 10
236 40
271 65
0 90
326 110
447 140
471 170

PROPS LABEL INDEX FROM-TO LENGTH PROPERTIES

114 5 57 58 28 INSTACK
121 60 61 106 132B
125 20 J 64 90 74B
276 90 121 127 16B
303 80 J 122 126 5B
332 160 J 141 180 133B
333 120 142 165 75B
EXT REFS NOT INNER
EXT REFS
NOT INNER
INSTACK
EXT REFS NOT INNER
EXT REFS

STATISTICS
PROGRAM LENGTH
5200CF 4 USED
704N 452

```

1      CCC
2      CCC
3      CCC
4      CCC
5      CCC
6      SUBROUTINE CMHATS (A, IROWA, ICOLA, MTA, N, IROWB, ICOLB, NTB, C)
7      C*****
8      CMHATS - GENERAL MATRIX MULTIPLY
9      AND
10     TRANSPOSE
11     SINGLE PRECISION VERSION
12
13     PERFORMS
14     WHEN
15     A * B - C MTA=0 NTB=0
16     A * B TRANSPOSE - C MTA=0 NTB=1
17     A TRANSPOSE * B - C MTA=1 NTB=0
18     A TRANSPOSE * B TRANSPOSE - C MTA=1 NTB=1
19
20     C - IS A MATRIX (ROWA) ROWS BY (COLA) COLUMNS
21     B - IS A MATRIX (ROWB) ROWS BY (COLB) COLUMNS
22     A, B AND C ARE STORED BY ROWS (EXAMPLE)
23
24     MATRIX STORED
25     A= 1 2 3 4 5 6
26         1 2 3 4 5 6
27         1 2 3 4 5 6
28         1 2 3 4 5 6
29         1 2 3 4 5 6
30         1 2 3 4 5 6
31
32     C*****
33     C*****
34
35     IF MTA .LT. 0, C IS NOT ZEROED OUT. HENCE THE ROUTINE, IN THIS
36     CASE, COMPUTES A * B + D - C WHERE THE MATRIX D HAS BEEN
37     STORED ROW-WISE AT C BY THE CALLING PROGRAM. IF MTA = -1, A
38     IS TRANSPOSED. IF MTA = -2, A IS NOT TRANSPOSED. NTB IS
39     DEFINED AS ABOVE AND IS INDEPENDENT OF MTA.
40
41     INTEGER ROWA, COLA, ROWB, COLB
42
43     DIMENSION A(1), B(1), C(1), IPARM(2)
44
45     ROWA = IROWA
46     COLA = ICOLA
47     ROWB = IROWB
48     COLB = ICOLB
49     MTA = IARS(MTA)
50
51     IF (MTA .EQ. (-2)) NTB = 0
52     IF (NTA .EQ. 0) .AND. NTB .EQ. 0 IF (COLA - ROWB) RO, 5, RO
53     IF (NTA .EQ. 1) .AND. NTB .EQ. 0 IF (ROWA - ROWB) RO, 5, RO
54     IF (NTA .EQ. 0) .AND. NTB .EQ. 1 IF (COLA - COLB) RO, 5, RO
55     IF (NTA .EQ. 1) .AND. NTB .EQ. 1 IF (ROWA - COLB) RO, 5, RO

```

11/03/82 20.07.46

FTN 4.8+564

SUBROUTINE CHMATS 74/74 OPT=1

```

5 IF (NTA .EQ. 1) GO TO 10
  ILM= ROWA
  KLM= COLA
  INCI= COLA
  INCKA= 1
  GO TO 20
10 ILM= COLA
  KLM= ROWA
  INCI= 1
  INCKA= COLA
20 IF (NTB.EQ.1) GO TO 30
  JLM= COLR
  INCJ= 1
  INCKB= COLB
  GO TO 40
30 JLM= ROWB
  INCJ= COLB
  INCKB= 1
40 IF (MTA .LT. 0) GO TO 47
  LIM = ILM * JLM
  DO 45 I = 1,LIM
45 C(I) = 0.0
47 IJ = 0
  I = 0
50 I = 1 + 1
  IFIX=1*INCI-COLA
  J = 0
60 J = J + 1
  IJ=IJ+1
  IA=IFIX
  JB=J*INCJ-COLR
  K = 0
70 K = K + 1
  IA=IA+INCKA
  JB=JB+INCKB
  C(IJ)=C(IJ)+ A(IA) * B(JB)
  IF (K .LT. KLM) GO TO 70
  IF (J .LT. JLM) GO TO 60
  IF (I .LT. ILM) GO TO 50
  RETURN
80 IPARM(1) = NTA
  IPARM(2) = NTB
  CALL MESSAGE (-30,21,IPARM(1))
  RETURN
  END

```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 CHMATS

SUPROUTINE INVERS 74/74 OPT=1

```

50 A(ICOLUMN,L) = SWAP
   IF(M.LE.O) GO TO 70
   DO 60 L=1,M
     SWAP = B(IROW,L)
     B(IROW,L) = B(ICOLUMN,L)
     B(ICOLUMN,L) = SWAP
60 B(ICOLUMN,L) = SWAP
C
65 C   DIVIDE PIVOT ROW BY PIVOT ELEMENT
C
70 PIVOT = A(ICOLUMN,ICOLUMN)
   DETERM = DETERM * PIVOT
   A(ICOLUMN,ICOLUMN) = 1.0E0
   DO 80 L=1,M
     A(ICOLUMN,L) = A(ICOLUMN,L) / PIVOT
   IF(M.LE.O) GO TO 100
   DO 90 L=1,M
     B(ICOLUMN,L) = B(ICOLUMN,L) / PIVOT
90 B(ICOLUMN,L) = B(ICOLUMN,L) / PIVOT
C
75 C   REDUCE NON PIVOT ROWS
C
100 DO 130 LI=1,M
   IF(LI.EQ.ICOLUMN) GO TO 130
   T = A(LI,ICOLUMN)
   A(LI,ICOLUMN) = 0.0F0
   DO 110 L=1,M
     A(LI,L) = A(LI,L) - A(ICOLUMN,L) * T
   IF(M.LE.O) GO TO 130
   DO 120 L=1,M
     B(LI,L) = B(LI,L) - B(ICOLUMN,L) * T
120 B(LI,L) = B(LI,L) - B(ICOLUMN,L) * T
130 CONTINUE
C
90 C   INTERCHANGE COLUMNS
C
DO 150 I=1,M
  L = N + 1 - I
  IF(INDEX(L,1).EQ.INDEX(L,2)) GO TO 150
  JROW = INDEX(L,1)
  JCOLUMN = INDEX(L,2)
  DO 140 K=1,M
    SWAP = A(K,JROW)
    A(K,JROW) = A(K,JCOLUMN)
    A(K,JCOLUMN) = SWAP
140 CONTINUE
150 CONTINUE
DO 170 K=1,M
  IF(INDEX(K,3).EQ.1) GO TO 160
  ISING = 2
  GO TO 180
160 CONTINUE
170 CONTINUE
  ISING = 1
180 RETURN
190 ISING = 2
  RETURN
  END

```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 INVERS

VARIABLES	SN	TYPE	RELOCATION
0 A	REAL	ARRAY	F.P.
0 B	REAL	ARRAY	F.P.
260 I	INTEGER		
0 INDEX	INTEGER	ARRAY	F.P.
0 ISING	INTEGER		F.P.
266 JCOLUMN	INTEGER		
261 K	INTEGER		
264 L1	INTEGER		
0 M	INTEGER		F.P.
263 PIVOT	REAL		
267 T	REAL		

267	ANAX	REAL	
0	DETERM	REAL	F.P.
266	ICOLDM	INTEGER	
265	IROW	INTEGER	
257	J	INTEGER	
265	JROW	INTEGER	
262	L	INTEGER	
0	N	INTEGER	F.P.
0	NDIM	INTEGER	F.P.
267	SWAP	REAL	

INLINE FUNCTIONS TYPE ARCS
ABS REAL 1 INTRIN

STATEMENT LABELS

0	10	0	20	INACTIVE	46	30
31	40	0	50		0	60
115	70	0	80		0	90
146	100	0	110		0	120
205	130	0	140		234	150
245	160	0	170		251	180
252	190					

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES	EXITS	NOT INNER
21	10	J	30 31	2R	INSTACK	12	NOT INNER
25	130	I	32 R7	165B			
27	40	J	37				
32	70	F	39 45	16R	OPT		
73	50	L	55 58	4R	INSTACK		
110	60	L	60 63	4R	INSTACK		
130	80	L	70 71	3R	INSTACK		
142	90	L	73 74	3R	INSTACK		
147	130	L1	78 R7	41B			
163	110	L	82 R3	4R	INSTACK		NOT INNER
200	120	L	85 86	4R	INSTACK		
213	150	I	91 101	24B			NOT INNER
230	140	K	96 100	3B	INSTACK		
240	170	K	102 107	10B	OPT		EXITS

STATISTICS

PROGRAM LENGTH	340R	224
57000R CM USED		


```

1      CCC
      CCC
      CCC
      CCC
      CCC
      SUBROUTINE COPEN(DB, BUFFER, NUM)
      INTEGER DB
      WRITE(6,910)DB
      FORMAT(IX,"OPENED DATABLOCK:",I5)
      RETURN
      END
10

```

SYMBOLIC REFERENCE MAP (R-1)

ENTRY POINTS
3 COPEN

VARIABLES	SN	TYPE	RELOCATION	INTEGER	F.P.
0 BUFFER		REAL	*UNUSED	0 DB	
0 NUM		INTEGER	*UNUSED		

FILE NAMES
TAPE6 FMT

STATEMENT LABELS
14 910 FMT

STATISTICS
PROGRAM LENGTH 208 16
52000 CH USED

```

1      CCC
      CCC
      CCC
      CCC
      CCC
      CCC
      SUBROUTINE CLOSE(DR,NUM)
      INTEGER DR
      WRITE(6,920)DR
920    FORMAT(IX,'CLOSED DATABLOCK: ',I5)
      RETURN
      END
10

```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 CLOSE

VARIABLES	SN	TYPE	RELOCATION	F.P.	0	NUM	INTEGER	*UNUSED	F.P.
DB	0	INTEGER							

FILE NAMES
TAPF6 FMT

STATEMENT LABELS
14 920 FMT

STATISTICS
PROGRAM LENGTH 208 16
52000 CH USED

```

1      CCC
      CCC
      CCC
      CCC
      CCC

5      SUBROUTINE PACK(ZN,DB,TRAILER)
      COMMON/PACKX/TYPIN,TYPOUT,IROW,NROW,INCR
      INTEGER TRAILER(7),DB
      DIMENSION ZN(1)
      WRITE(6,100) DB
      FORMAT(1X,"DATABASE=",15)
      NWDPLN=10
      J=0
      LIMIT=NROW/NWDPLN
      IF(LIMIT.LE.0)GOTO 200
      DO 300 I=1,LIMIT
        WRITE(6,110)(ZN(J+K),K=1,NWDPLN)
        J=J+NWDPLN
      CONTINUE
      300
      200 LIMIT=NROW-LIMIT+NWDPLN
      IF(LIMIT.LE.0)GOTO 400
      WRITE(6,110)(ZN(J+1),I=1,LIMIT)
      400 RETURN
      110 FORMAT(10(1X,E12.5))
      25  END
  
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 PACK

VARIABLES	SN	TYPE	RELOCATION	107	I	INTEGER	PACKX
0 DB	1	INTEGER	F.P.	107	1	INTEGER	PACKX
4 INCR	2	INTEGER	PACKX	2	IROW	INTEGER	PACKX
105 J	110	INTEGER		110	K	INTEGER	PACKX
106 LIMIT	3	INTEGER		3	NROW	INTEGER	PACKX
104 NWDPLN	0	INTEGER		0	TRAILER	INTEGER	F.P.
0 TYPIN	1	REAL	PACKX	1	TYPOUT	REAL	PACKX
0 ZN		REAL	ARRAY				

FILE NAMES
TAPES
FMT

STATEMENT LABELS
62 100 FMT
0 100

LOCUS	LABEL	INDEX	PROG-TC	LENGTH	PROPERTIES	EXT REFS	NOT INNER
14	100	1	16 19	20P			
21		4	17 17	10R			
22		1	27 22	10P			

36 200

PAGE 2

11/03/82 20.07.46

FTN 4.84564

74/74 OPT-1

SUBROUTINE PACK
COMMON BLOCKS LENGTH
PAGE 5

STATISTICS
PROGRAM LENGTH 114R 76
CM LABELED COMMON LENGTH 5B 5
52000B CM USED

```

1      CCC
      CCC
      CCC
      CCC
      CCC
      CCC

5      SUBROUTINE RDTL(TRACE)
      COMMON/DTRACE/TACT(6),TACPA(6),TTOSEL(6),TCPLA(6),TSIL(6),
      ITCTA(6)
      INTEGER TRACE(7),TACT,TACPA,TTOSEL,TCPLA,TSIL,TECTA
      COTO(100,200,300,400,500,600),TRAIL(1)-100
      DO 10 I=1,6
      TRAIL(I+1)=TACT(I)
      RETURN
      DO 20 I=1,6
      TRAIL(I+1)=TTOSEL(I)
      RETURN
      DO 30 I=1,6
      TRAIL(I+1)=TECTA(I)
      RETURN
      DO 40 I=1,6
      TRAIL(I+1)=TCPLA(I)
      RETURN
      DO 50 I=1,6
      TRAIL(I+1)=TACPA(I)
      RETURN
      DO 60 I=1,6
      TRAIL(I+1)=TSIL(I)
      RETURN
      END

```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 RDTL

VARIABLES	SN	TYPE	RELOCATION	0	TACT	INTEGER	ARRAY	DTRACE	DTRACE
72 I		INTEGER							
6 TACPA		INTEGER	ARRAY	DTRACE					
27 TCPLA		INTEGER	ARRAY	DTRACE					
30 TSIL		INTEGER	ARRAY	DTRACE					

STATEMENT LABELS

0 10	0 20	0 30
0 40	0 50	0 60
20 100	27 200	36 300
45 400	54 500	63 600

LOOP LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
23 10	1	11 12	2R	INSTACK
32 20	1	14 15	2R	INSTACK
41 30	1	17 18	2R	INSTACK
50 40	1	20 21	2R	INSTACK
59 50	1	23 24	2R	INSTACK

PROPERTIES
INSTACK

OPT-1
LENGTH 28

74/74
FROM-TO 26 27

SUBROUTINE RDTBL
LABEL INDEX
66 60 1

COMMON BLOCKS
DTRAIL 36

STATISTICS

PROGRAM LENGTH 738 59
CM LABELED COMMON LENGTH 448 36
520008 CM USED

```

1      CCC
      CCC
      CCC
      CCC
      CCC
5
      SUBROUTINE WRTTL(MC)
      INTEGER MC(7)
      WRITE(6,100)
      FORMAT(IX,'WRTTL SAID MCR--')
      DO 200 I=1,7
      100  WRITE(6,110)MC(I)
      110  FORMAT(3X,18)
      RETURN
      END

```

SYMBOLIC REFERENCE MAP (R-1)

ENTRY POINTS
3 WRTTL

VARIABLES	SM	TYPE	RELOCATION	0	MC	INTEGER	ARRAY	F.P.
36 I		INTEGER						

FILE NAMES MONF
TAPPA FMT

STATEMENT LABELS
24 100 FMT

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES	EXT	REFS
11	200	I	10 11	108			

STATISTICS
PROGRAM LENGTH 418 33
570008 CM USED

```

1      CCC
      CCC
      CCC
      CCC
      CCC
      CCC

5      SUBROUTINE READ(DB,ZN,MV,FLAG,M)
      COMMON/ZDATA/ZACPT(49),ZGCPA(164),ZTOSEL(4),ZCPLA(41),ZSIL(18),
      IZECTA(64,4)
      INTEGER DB,FLAG,ZTOSEL,ZCPLA,ZSIL,ZECTA,ZNI(41)
      REAL ZACPT,ZGCPA,ZN(144),ZNR(41)
      EQUIVALENCE(ZNI(1),ZNR(1))
      DATA IACPT,ITOSEL,JECTA,JECTA,ICPLA,ICGPA,ISIL/1,1,1,1,1,1,1/
      READS DATA FROM VARIOUS ARRAYS INTO THE Z VECTOR
      GOTO(10,20,30,40,50,60),DB-100
      READ FROM ZACPT
      DO 200 I=1,MV
      IF(IACPT+I-1.GT.49)GOTO 300
      ZN(I)=ZACPT(IACPT+I-1)
      IACPT=IACPT+MV
      M=MV
      IF(FLAG.NE.1)RETURN
      IACPT=50
      RETURN
      M=I-1
      IACPT=50
      RETURN

300     READ FROM ZTOSEL
      DO 210 I=1,MV
      IF(ITOSEL+I-1.GT.4)GOTO 320
      ZNI(I)=ZTOSEL(ITOSEL+I-1)
      ZNI(I)=ZNR(I)
      ITOSEL=ITOSEL+MV
      M=MV
      IF(FLAG.NE.1)RETURN
      ITOSEL=5
      RETURN
      M=I-1
      ITOSEL=5
      RETURN

320     READ FROM ZECTA
      DO 220 I=1,MV
      IF(ZECTA(JECTA+I-1,JECTA).EQ.-1)GOTO 340
      ZNI(I)=ZECTA(JECTA+I-1,JECTA)
      ZNR(I)=ZNR(I)
      M=MV
      IECTA=JECTA+MV
      IF(FLAG.NE.1)RETURN
      IECTA=1
      IECTA=JECTA+1
      RETURN
      M=I-1
      IECTA=1
      IECTA=JECTA+1

340

```

```

SUBROUTINE READ      74/74  OPT=1
IF(JECTA.LE.4)RETURN
JECTA=4
LECTA=64
RETURN
60      READ FROM CPLA
      DO 240 I=1,NW
      IF(1CPLA+I-1.GT.41)GOTO 360
      ZN(I)=2CPLA(1CPLA+I-1)
      ZN(I)=ZNR(I)
      1CPLA=1CPLA+NW
      M=NW
      IF(FLAG.NE.1)RETURN
      1CPLA=42
      RETURN
      360      M=1-1
      1CPLA=42
      RETURN
70      READ FROM BCPLA
      DO 250 I=1,NW
      IF(1BCPLA+I-1.GT.164)GOTO 380
      ZN(I)=2BCPLA(1BCPLA+I-1)
      1BCPLA=1BCPLA+NW
      M=NW
      IF(FLAG.NE.1)RETURN
      1BCPLA=165
      RETURN
      380      M=1-1
      1BCPLA=165
      RETURN
90      READ FROM SIL
      DO 260 I=1,NW
      IF(1SIL+I-1.GT.18)GOTO 390
      ZN(I)=2SIL(1SIL+I-1)
      ZN(I)=ZNR(I)
      1SIL=1SIL+NW
      M=NW
      IF(FLAG.NE.1)RETURN
      1SIL=19
      RETURN
      390      M=1-1
      1SIL=19
      RETURN
      END
105

```

SYMBOLIC REFERENCE MAP (R=1)

PTN 4.8+564 11/03/82 20.07.46

74/74 OPT-1

SUBROUTINE READ

ENTRY POINTS

3 READ

VARIABLES SN TYPE RELOCATION F.P.

0 DB	INTEGER		0	FLAC		
221 I	INTEGER		212 IACPT			
217 IBCPA	INTEGER		214 IECTA			
216 ICPLA	INTEGER		220 ISIL			
213 ITOSL	INTEGER		215 JECTA			
0 W	INTEGER		0 WU			
0 ZACPT	REAL		61 ZBCPA			
424 ZECTA	INTEGER		331 ZCPLA			
0 ZN	REAL		222 ZMI			
222 ZMR	REAL		402 ZSIL			
325 ZTOSL	INTEGER					

STATEMENT LABELS

21 10	44 20
117 40	143 50
0 200	0 210
0 240	0 250
40 300	64 320
137 360	162 380

LOOPS LABEL INDEX FROM-TO LENGTH PROPERTIES

22 200	1	16 18	108	OPT	EXITS
45 210	1	30 33	108	OPT	EXITS
71 220	1	45 48	108	OPT	EXITS
120 240	1	65 68	108	OPT	EXITS
144 250	1	80 82	108	OPT	EXITS
167 260	1	94 97	108	OPT	EXITS

COMMON BLOCKS LENGTH

ZDATA

532

STATISTICS

PROGRAM LENGTH	3108	200
CM LABELED COMMON LENGTH	10248	532
520000P CM USED		

DTI	BCPA	0	41	23	0	0	0	0+RC	0
+RC	0ENDREC							+RC	1
DTI	BCPA			1				0+RC	2
+RC	2	0	-225000000E+01					0+RC	3
+RC	3	0	-225000000E+01					0+RC	4
+RC	4	0	-300000000E+01					0+RC	5
+RC	5	0	-300000000E+01					0+RC	6
+RC	6	0	-225000000E+01					0+RC	7
+RC	7	0	-225000000E+01					0+RC	8
+RC	8	0	-225000000E+01					0+RC	9
+RC	9	0	-225000000E+01					0+RC	10
+RC	10	0	-225000000E+01					0+RC	11
+RC	11	0	-225000000E+01					0+RC	12
+RC	12	0	-225000000E+01					0+RC	13
+RC	13	0	-225000000E+01					0+RC	14
+RC	14	0	-225000000E+01					0+RC	15
+RC	15	0	-225000000E+01					0+RC	16
+RC	16	0	-225000000E+01					0+RC	17
+RC	17	0	-225000000E+01					0+RC	18
+RC	18	0	-225000000E+01					0+RC	19
+RC	19	0	-225000000E+01					0+RC	20
+RC	20	0	-225000000E+01					0+RC	21
+RC	21	0	-225000000E+01					0+RC	22
+RC	22	0	-225000000E+01					0+RC	23
+RC	23	0	-225000000E+01					0+RC	24
+RC	24	0	-225000000E+01					0+RC	25
+RC	25	0	-225000000E+01					0+RC	26
+RC	26	0	-225000000E+01					0+RC	27
+RC	27	0	-225000000E+01					0+RC	28
+RC	28	0	-225000000E+01					0+RC	29
+RC	29	0	-225000000E+01					0+RC	30
+RC	30	0	-225000000E+01					0+RC	31
+RC	31	0	-225000000E+01					0+RC	32
+RC	32	0	-225000000E+01					0+RC	33
+RC	33	0	-225000000E+01					0+RC	34
+RC	34	0	-225000000E+01					0+RC	35
+RC	35	0	-225000000E+01					0+RC	36
+RC	36	0	-225000000E+01					0+RC	37
+RC	37	0	-225000000E+01					0+RC	38
+RC	38	0	-225000000E+01					0+RC	39
+RC	39	0	-225000000E+01					0+RC	40
+RC	40	0	-225000000E+01					0+RC	41
+RC	41	0	-225000000E+01					0+RC	42
+RC	42	0	-225000000E+01					0+RC	43

DTI	SIL	0	1A	10A	0	0	0	0+SI	0
+SI	0ENDREC							+SI	1
DTI	SIL			7	13			+SI	2
+SI	2	37	43	55	61			+SI	3
+SI	3	85	91	97	103			+SI	4

Appendix B

```

ID SAMPLE
APP DMAP
BEGIN $
$ SETS UP THE TABLES THAT DESCRIBE THE STRUCTURAL MODEL
$
GPI GEOM1,GEOM2,/GPL,EQEXIN,GPD1,CSTM,BGPD1,SIL/V,N,LUSET/ V,N,
MOCPT $
SAVE LUSET,MOCPT $
COND ERROR1,MOCPT $
CP2 GEOM2,EQEXIN/ECT $
PARAML PCDB,/C,N,PRES/C,N,/C,N,/C,N,/V,N,JHPPLOT $
CP3 GEOM3,EQEXIN,GEOM2,/CPT/V,N,MOCRAV $
TAI ECT,EPT,BGPD1,SIL,GPT,CSTM/EST,CEI,GPECT,/V,N,LUSET/ V,N,
MOSIMP/C,N,1/V,N,MOCENL/V,N,GENEL $
SAVE MOCENL,MOSIMP,GENEL $
COND ERROR1,MOSIMP $
PARAM //C,N,ADD/V,N,MOCGX/C,N,1/C,N,0 $
EMC //C,N,ADD/V,N,MOCGX/C,N,1/C,N,0 $
EST,CSTM,MPT,DIT,GEOM2,/KELM,KDICT,MELM,MIDICT,,/V,N,MOCGX/ V,
N,MOCGX/C,N,/C,N,/C,N,/C,N,/C,Y,COUPHASS/C,Y,CPBAR/C,Y,CPROD/ C,Y,
CPQAD1/C,Y,CPQAD2/C,Y,CPTRIA1/C,Y,CPTRIA2/C,Y,CPTUBE/ C,Y,
CPQDPLT/C,Y,CPTMPLT/C,Y,CPTBSC $
MOCGX,MOCGX $
SAVE JMPCGX,MOCGX $
COND IMA GPECT,KDICT,MELM/KCGX,GPT $
LABEL JMPCGX $
COND ERROR1,MOCGX $
EMA GPECT,KDICT,MELM/MCG,/C,N,-1/C,Y,WTMASS=1.0 $
COND LGPWC,CRDPT $
CPMG BGPD1,CSTM,EQEXIN,MCG/OGPWG/V,Y,CRDPT=-1/C,Y,WTMASS $
OFF OGPGC,..../V,N,CARDNO $
LABEL LGPWC $
EQUIV KCGX,KCG/MOCENL $
COND LRL11,MOCENL $
SHA3 CEI,/KCGX/V,N,LUSET/V,N,MOCENL/C,N,-1 $
ADD KCGX,KCGY/KCG $
LABEL LRL11 $
PARAM //C,N,MPT/V,N,NSKIP/C,N,0/C,N,0 $
PARAM //C,N,SUB/V,N,DESINT/C,N,0/C,N,1 $
PURGE WRRR,DIJE,D2JE/DESINT $
PARAM //C,N,MPT/V,N,MOCAMBER/C,N,1/C,Y,ICAHB=-1 $
PURGE CAMBER/MOCAMBER $
$ THE CONSTRAINTS OF THE FIRST SURCASE ARE APPLIED
$
GPI CASECC,GEOM4,EQEXIN,GPD1,BGPD1,CSTM/RC,LUSET,ASET/V,N,LUSET/
V,N,MPCF1/V,N,MPCF2/V,N,SINGLE/V,N,OHIT/V,N,REACT/V,N,NSKIP/V,
K,REPEAT/V,N,MOCSET/V,N,MOL/V,N,MOA/C,Y,SUBID $
MPCF1,MPCF2,SINGLE,OHIT,REACT,NSKIP,REPEAT,MOCSET,MOL,MOA $
GPI,GPT,LUSET,SIL/OGPST/V,N,MOCIST $
SAVE MOCIST $
COND LRL4,MOCIST $
OFF OGPGC,..../V,N,CARDNO $
LABEL IRL4 $
EQUIV OGPGC,MPCF1/MCG,MAN/MPCF1 $
COND LRL2,MPCF1 $

```

[illegible]


```
CASECC,UCV,KELM,KDICT,ECT,EQEXIN,CPECT,PGC,QC/QMRGYI,OCPPBI/
C.N.STATICS $
ONRGYI,OCPPBI,...// $
CASECC,CSHM,NPT,DIT,EQEXIM,SIL,GTTT,EDT,BGPDOT,QC,UCV,EST,
XYCDB,PGC/OPCI,OQC1,OUCVI,OES1,OEFL,PUCVI/C.M.STATICS/V,M,
NOSORT2--1 $
NOSORT2 $
LBL21,NOSORT2 $
OUCVI,OPCI,OQC1,OEFL,OES1,/OHCV2,OPG2,OQC2,OEFL,OES2, $
OUCV2,OPG2,OQC2,OEFL,OES2,//V,M,CARDNO $
CARDNO $
XYCDB,OPG2,OQC2,OUCV2,OES2,OEFL/XYPLTT/C,M,TRAN/C,M,PSET/V,M,N,
PFLE/V,M,CARDNO $
XYPLTT// $
DPLDT $
LBL21 $
OUCVI,OPCI,OQC1,OEFL,OES1,//V,M,CARDNO $
CARDNO $
P2,JUMPPLOT $
DPLDT $
PLTPAR,GPSETS,ELSETS,CASECC,BGPDOT,EQEXIN,SIL,PUCVI,CPECT,OESI/
PLOTXZ/V,M,NBIL/V,M,LUSET/V,M,JUMPPLOT/V,M,PLTLG/V,M,PFLE $
PFLE $
PLOTXZ// $
FINIS $
ERROR2 $
//C,M,-2/C,M,FLUTTER $
ERROR1 $
//C,M,-1/C,M,FLUTTER $
ERROR4 $
//C,M,-4/C,M,FLUTTER $
ERROR9 $
//C,M,-3/C,M,STATICS $
FINIS $
END $
TIME .0
DIAG 14
COND
$
$
$
TITLE-SAMPLE
METHOD-10
ECHO-POIN
DISPLACEMENT - ALL
SUBSEQUENCE TO SET INITIAL ANGLE OF ATTACK
SUBCASE 1
SPC = 1
SUBSEQUENCE TO CALCULATE REFLECTIONS, ETC.
SUBCASE 2
SPC = 2
```

```

INPUT BULK DATA DECK ECHO
1 2 3 4 5 6 7 8 9 10
$ TOLERANCE OF ITERATION LOOP
PARAM ELOOP .01
$ ROOT CHORD
PARAM RC 40.0
$ WING PLANFORM AREA, AERO MODEL
PARAM WAREA 2100.0
$ DYNAMIC PRESSURE
PARAM Q 59.34009
$ ANGLE OF ATTACK PARAMETER
PARAM AOAP 1.134925 0.0
$ PARAMETER TO INCLUDE CAMBER MATRIX IN CALCULATIONS
PARAM ICAMB 1
$ PARAMETER TO PICK WHICH EIGENVECTOR TO USE
PARAM LHODES 1
$
$ CONSTRAINTS FOR BOTH SUBCASES
SPC1 1 2 1 THRU 6
SPC1 1 13 5
SPC1 1 456 1 THRU 18
SPC1 2 123 1 THRU 6
SPC1 2 456 1 THRU 18
$ DOF FOR ANALYSIS SET
ASET1 3 7 THRU 18
$
$
$ DMI CAMBER 0 2 1 1.3123 17 1
DMI CAMBER 1 15 1.3123 17 1.3123
$
$ AERODYNAMIC PARAMETERS
AERO 223.4 40.0 7.650-2 1
HKAERO1 .2
+FAERO1 1.0-15
$
$ DESCRIBES DIMENSIONS OF AERO MODEL
CAERO1 1000 1 2 610
+FAERO1 0.0 2.25 40. 11.25 45. 2.25 2.25 1
CAERO1 1050 1 1 1
+FAERO1 11.25 45. 2.25 16.25 15. 60. 2.25 15.
CAERO1 1075 1 1 1
+FAERO1 27.5 45. 2.25 16.25 30. 60. 2.25 15.
AERFACT 610 .0 .66666671.0
FAERO1 1
$
$
$ COORDINATE
COORD1 1 1 1 3 9 7
COORD2 2 2 3 5 11 9
COORD3 3 3 7 9 15 13

```

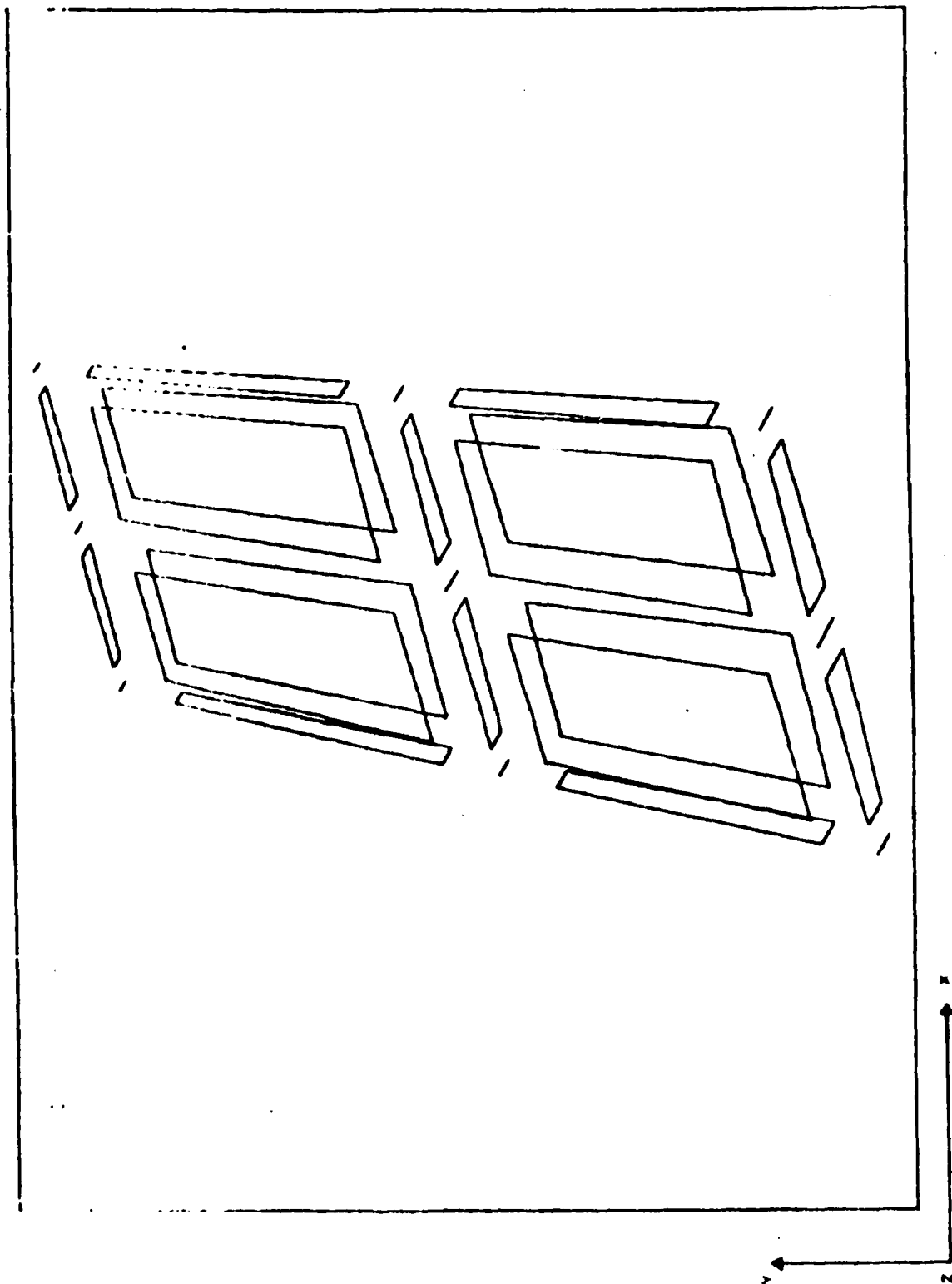


```

INPUT BULK DATA DECK ECHO
. 1 .. 2 .. 3 .. 4 .. 5 .. 6 .. 7 .. 8 .. 9 .. 10 .
PQDNEM2 1 1.1
PQDNEM2 2 1.1
PQDNEM2 3 1.1
PQDNEM2 4 1.1
PQDNEM2 5 1.1
PQDNEM2 6 1.1
PQDNEM2 7 1.1
PQDNEM2 8 1.1
PROD 1 1.1
PROD 2 1.1
PROD 3 1.1
PROD 4 1.1
PROD 5 1.1
PROD 6 1.1
PROD 7 1.1
PROD 8 1.1
PROD 9 1.1
PSHEAR 1 1 .06
PSHEAR 2 1 .06
PSHEAR 3 1 .06
PSHEAR 4 1 .06
PSHEAR 5 1 .06
PSHEAR 6 1 .06
PSHEAR 7 1 .06
PSHEAR 8 1 .06
PSHEAR 9 1 .06
PSHEAR 10 1 .06
PSHEAR 11 1 .06
$
$
$ DEVINES THE TRANSFORMATION MATRIX FROM STRUCTURAL DEFORMATIONS
$ TO AERO DISPLACEMENTS
SET1 1 1 3 5 7 9 11 13 +S1
+S1 15 17
SPLINE1 2000 1000 1000 1003 1 0.0
SPLINE1 3000 1050 1050 1050 1 0.0
SPLINE1 4000 1075 1075 1075 1 0.0
D-1 TOSEL 0 4
+DT11 ENDREC
$
$
$ TOP SURFACE ELEMENT LIST
DT1 TOSEL 1 1 2 3 4 ENDRLC
ENDDATA

```

c



REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GAE/AA/82D-16	2. GOVT ACCESSION NO. AD A724 662	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) STATIC AEROELASTIC ANALYSIS OF FLEXIBLE WINGS VIA NASTRAN		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kim Jones 1Lt.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1982
		13. NUMBER OF PAGES 130
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17. LYNN E. WOLAYER Dean for Research and Professional Development Air Force Institute of Technology (AFIT) Wright-Patterson AFB OH 45433 6 JAN 1983		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) NASTRAN Flexible-wings Aeroelastic Fluid-Structure Interfacing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The purpose of this study was to expand the capabilities of the Static Flexible Wing Aeroelastic Sequence that Captain Lance P. Chrisinger developed for NASTRAN as his Master's Thesis at AFIT. Captain Chrisinger developed a basic procedure to enable NASTRAN to analyze flexible wing airloads and stresses. That capability is expanded to enable analysis of standard wing models.		

END

FILMED

3-83

DTIC